Java 2: Ready for Prime Time?

# JAVA Pro™

www.java-pro.com
A Fawcette Technical Publication
February 1999, Vol. 3, No. 2

**REVIEWS**
- DashO-Pro 1.1
- jtest! 2.04
- JDesignerPro 3.0

8 Tips for Browser Compatibility

# Swing into Action

## Java's New Look and Feel

**CORBA and Java Head to Head**

**Faster Sorting with Collections**

**Network Programming with Datagrams**

**Build a What's New Servlet**

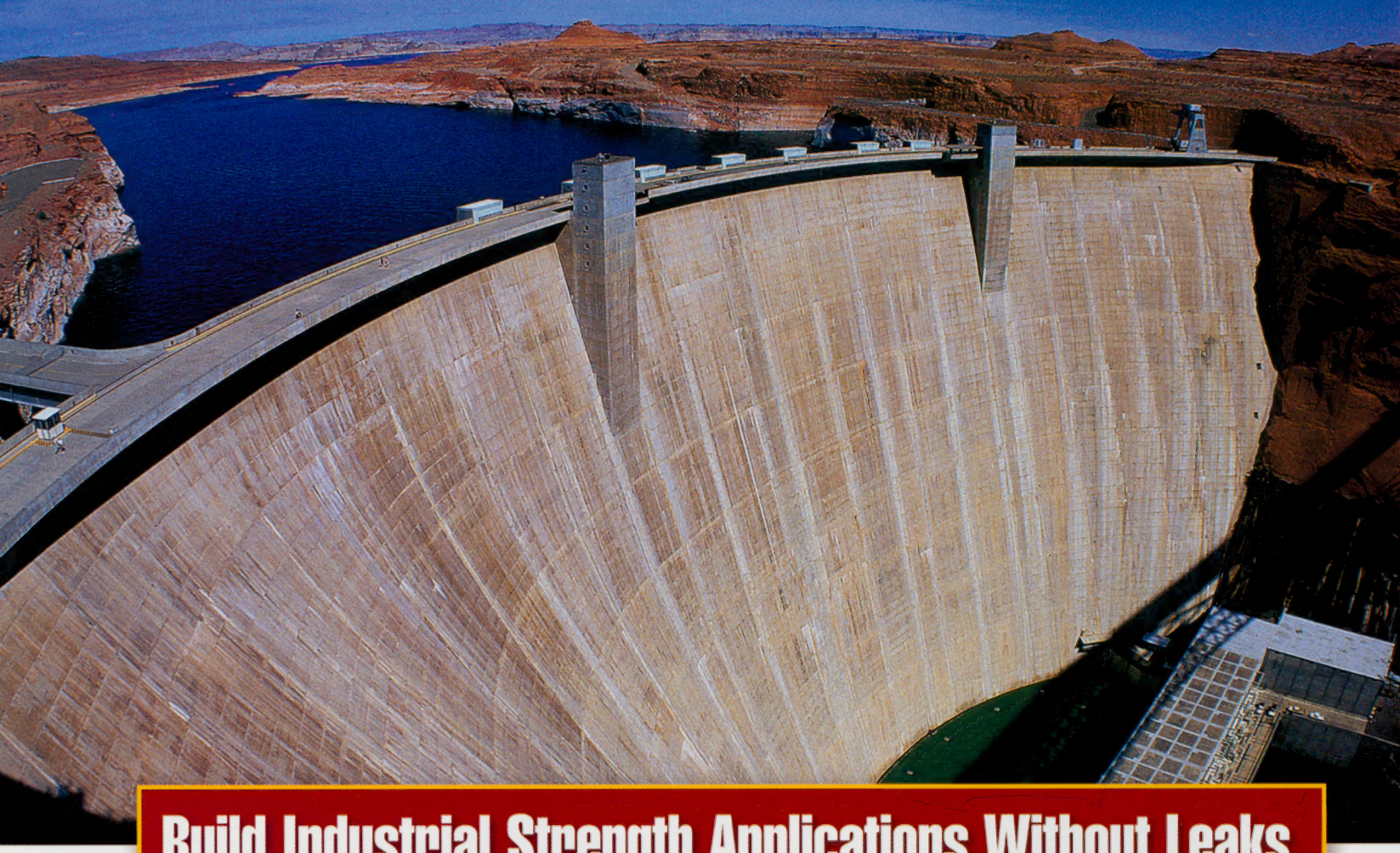
**Trapping Data Errors**

**Code Online**

**Free Source Code www.java-pro.com**

U.S. $5.95 (Canada $7.95)

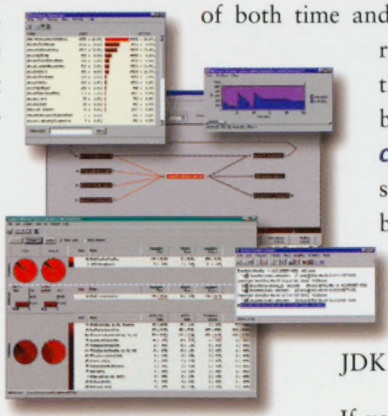
02>
7 25274 91607 8



# Build Industrial Strength Applications Without Leaks

Whether you're building a high performance hydro-electric dam or a ***MISSION-CRITICAL JAVA APPLICATION***, you know that those pesky leaks can sabotage everything! You need a professional tool that is accurate, easy to use and helps you deliver ***SCALABLE, ROBUST PERFORMANCE*** from start to finish.

***JPROBE PROFILER*** is used by Java developers around the world to improve the performance of their Java applications. And now, the most ***POWERFUL TOOL*** for finding and eliminating Java performance bottlenecks makes it easy for you to find and plug memory leaks in your Java code. Use the new ***HEAP REFERENCE GRAPH*** to instantly see a heap snapshot analysis and untangle even the most complex object references that prevent garbage collection.



JPROBE Profiler

JProbe Profiler gives you the most ***ACCURATE MEASUREMENT*** of both time and memory consumption while your program runs. It tracks everything for you, thread by thread, with method details shown clearly right back to each line of source code*. The advanced ***CALL GRAPH*** helps you analyze your profiling snapshot and quickly drill-down to performance bottlenecks. And you'll get superior performance where you want it because JProbe Profiler provides ***MULTI-PLATFORM*** support for Windows NT and Solaris in JDK1.1, JDK1.2, and JIT environments.

If you develop Java code for a living, you need a ***PROFESSIONAL JAVA PROFILER.*** Get a free evaluation copy of ***JPROBE PROFILER*** right now and start making your Java code industrial strength, without leaks.

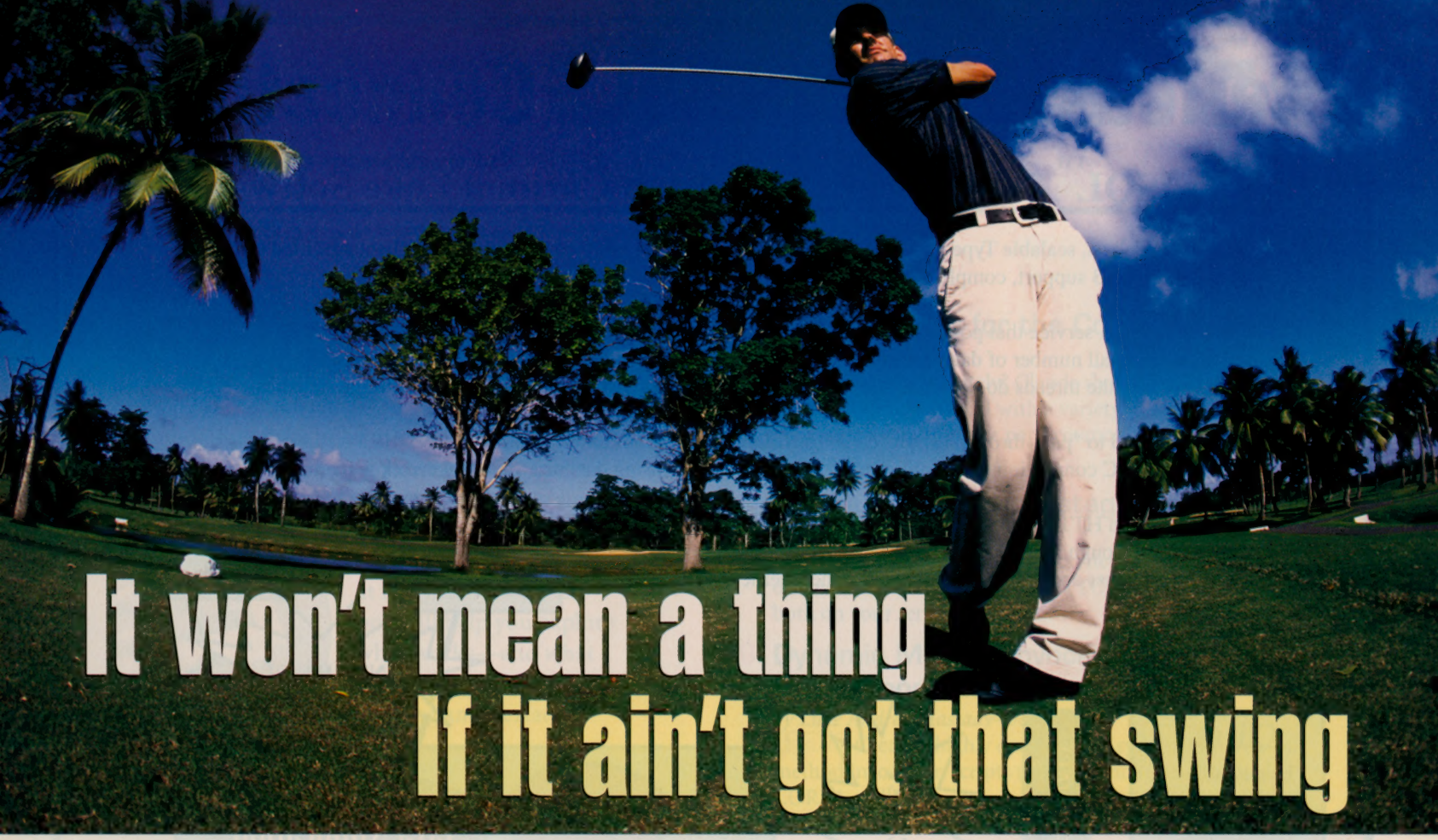
***Try JProbe Now! www.klg.com/jprobe*** | ***The Easiest Way to Search and Destroy Java Performance Bottlenecks***

KL GROUP
*Software Development Productivity*

KL Group Inc. Toll-Free: 1-888-328-9599 Tel: 416-594-1026 Fax: 416-594-1919
KL Group Europe B.V. Tel: +31 (0)20 679 95 03 Fax: +31 (0)20 470 03 26
www.klg.com

JAVA COMPATIBLE

* Source code not required to profile with JProbe. JProbe is a trademark of KL Group Inc. All other products are trademarks or registered trademarks of their respective companies. JP003



# It won't mean a thing
# If it ain't got that swing

## JCLASS JavaBeans

You need ***powerful components*** for professional Java development. But if they don't work with Swing and the latest JDK 1.2 release... they won't mean a thing.

JClass, the world's most popular suite of 100% Pure JavaBeans, now includes ***JClass SwingSuite***, the essential set of extensions and enhancements for Swing in JDK 1.2. And with the power of JClass Suites, you'll be building for the future ***today.***

**JCLASS WORKS WITH:**

| | | |
|---|---|---|
| ✓ JDK 1.0.2 (AWT) | ✓ IBM VisualAge for Java | ✓ Progress Apptivity |
| ✓ JDK 1.1 (AWT or Swing) | ✓ IST Visaj | ✓ SilverStream |
| ✓ JDK 1.2 (Swing) | ✓ Microsoft Visual J++ | ✓ Sun Java WorkShop |
| ✓ Bluestone Sapphire/Web | ✓ NetBeans | ✓ SuperCede |
| ✓ Borland JBuilder | ✓ Powersoft PowerJ | ✓ Symantec Visual Café |

New **JClass SwingSuite** - easily add the power of MDI, wizards and a robust collection of thread-friendly, lightweight components including JCTreeExplorer, JCTreeTable, JCSpinBoxes, a configurable DateChooser control and more!

- **JClass LiveTable** - the high performance Java grid/table for static, dynamic or db-bound data
- **JClass Chart** - the ultimate in Java charting and graphing, designed for superior performance
- **JClass HiGrid** - the innovative RAD outline-grid for hierarchical dynamic data
- **JClass DataSource** - a robust, hierarchical, multiple-platform data source
- **JClass Field** - complete input and validation for popular data types including calendar

Need rock-solid, robust Java GUIs? Professional Java developers around the world have made the award-winning JClass family of components their choice. Where else will you find seamless IDE integration and support for all major JDK versions? *Try JClass today for great performance with an improved swing!*

**EVALUATE/ORDER ONLINE** www.klgroup.com

KL GROUP
*Software Development Productivity*

KL Group Inc. Toll-Free: 1-888-328-9598 Tel: 416-594-1026 Fax: 416-594-1919
KL Group Europe B.V. Tel: +31 (0)20 679 95 03 Fax: +31 (0)20 470 03 26
www.klgroup.com

* Databinding feature requires purchase of JClass DataSource, or use of Borland JBuilder Professional or C/S edition, or Symantec Café dbDE edition. JClass, JClass Chart, JClass DataSource, JClass Field, JClass HiGrid, JClass LiveTable and JClass SwingSuite are trademarks of KL Group Inc. All other product names are trademarks or registered trademarks of their respective owners. JP004

# Serious "Java to SQL" Connectivity

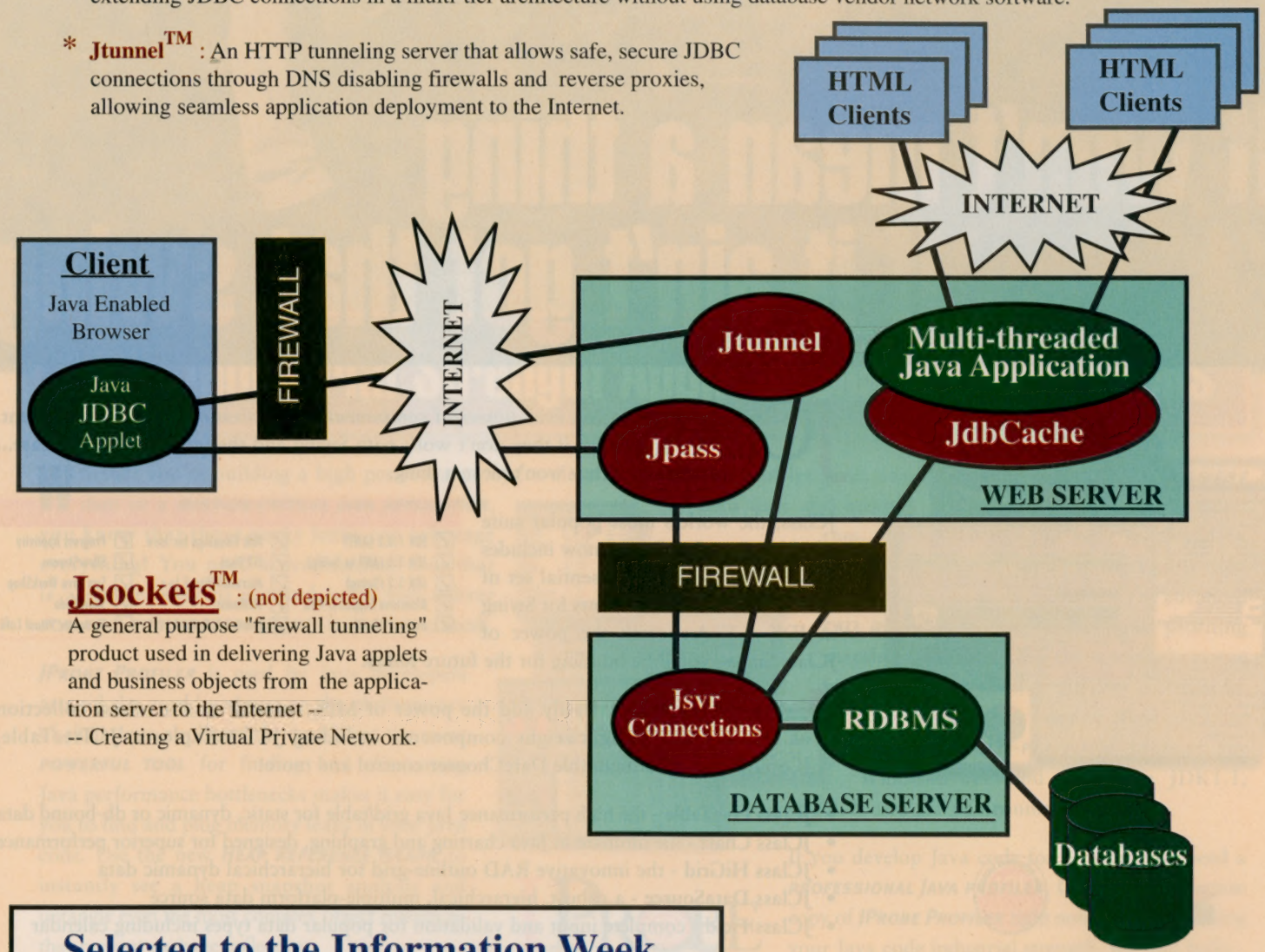
## The JSQL™ 3.0 Software Suite from Caribou Lake Software

* **Jsvr™**: A robust, scalable Type 3 JDBC connection server that is highly configurable with client specified time-outs, meta-data, BLOB support, complete connection logging and advanced security features.

* **JdbCache™**: A service that pools, reuses, and serves database connections to applications. This allows a Java application to use a small number of database connections to service many requests. It's ideal for multi-threaded Java programs because the threads do not have to create and use their own JDBC connection. Includes usage statistics.

* **JPass™**: Used to "pass-through" JDBC calls from client applications to any database server via a web server, hence extending JDBC connections in a multi-tier architecture without using database vendor network software.

* **Jtunnel™**: An HTTP tunneling server that allows safe, secure JDBC connections through DNS disabling firewalls and reverse proxies, allowing seamless application deployment to the Internet.




**Jsockets™**: (not depicted)
A general purpose "firewall tunneling" product used in delivering Java applets and business objects from the application server to the Internet --
-- Creating a Virtual Private Network.

**Selected to the Information Week Top 100 Internet Solution Series**
**(October 5, 1998)**

*Learn more about our products, free trial software, and superior professional service offerings at:*

**www.CaribouLake.com**

CARIBOU LAKE SOFTWARE

*Superior Java/ RDBMS/ Data Warehouse Products and Services*

*Minneapolis + Milwaukee + San Francisco*

JAVA Pro™

February 1999,
Vol. 3, No. 2

# Contents

## Features


### 20 A Test Panel for Pluggable Look and Feel

*by Richard G. Baldwin*

You no longer have to be ashamed of your Java GUIs. Swing provides you with a new set of options, including the ability to change your application's Look and Feel at run time. Dick's test bed will help you explore what works—and what doesn't—when you're designing an interface.

### 28 A SortFactory for the Collections API

*by Claude Duguay*

Feeling out of sorts lately? Claude shows you how to extend the new Collections API with a variety of sorting algorithms, allowing to you optimize performance on any list or array of objects.

### 36 Introduction to the Datagram Protocol

*by Richard G. Baldwin*

In the final article of our series on networking programming, discover how UDP complements TCP and how you can support both on your server applications.

### 46 Dynamic Method Invocation with CORBA and Java

*by Luke Andrew Cassady-Dorion*

Java 101 Create a highly configurable system with dynamic method invocation scripts using technologies provided by CORBA and Java.


## Reviews


10 DashO-Pro 1.1

12 jtest! 2.04

14 JDesignerPro 3.0


## Departments


5 Editor's Note
*by Tyler Sperry*

7 Letters

16 New Products

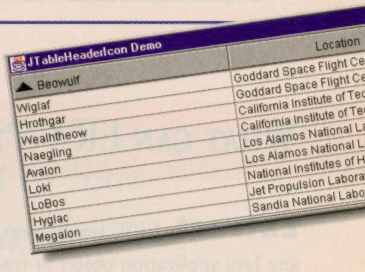
76 Ad Index


## Columns


### 54 Ask the Java Pro

*by Daniel Savarese*

The release of the Java 2 platform provides the opportunity to ask ourselves, "Are we there yet?"


To learn how to add a sort-direction icon to a column head, see page 54.


### 62 Javatecture

*by James W. Cooper*

Dirty data can cause application failure if you're not careful. James shows you how to use exceptions to keep your programs running smoothly.

### 66 Impure Thoughts

*by Rick Grehan*

Java 101 Priority inversion can occur in almost any multitasking system that supports task priorities. Learn how to keep your application's priorities in line.

### 70 Servlet Solutions

*by Larry O'Brien*

Web This month, learn how to create a password protected "What's New" page that's dynamically generated from a JDBC-accessed database.

### 78 VM Roadtest

*by Paul Tyma*

Getting your applets to run in IE and Navigator can cause headaches even for the most experienced developers. Paul provides eight tips for dealing with runtime idiosyncrasies.


*Java Pro* is published monthly for $34.97 per year, by Fawcette Technical Publications, 209 Hamilton Avenue, Palo Alto, CA, USA 94301-2500. Tel. 650-833-7100; Fax. 650-853-0230. **Postmaster:** Send address changes to Java Pro, P.O. Box 54584, Boulder, CO 80322-4584. Application to mail at Periodicals Postage Rates is pending at Palo Alto, CA and additional mailing offices. © **1999 Fawcette Technical Publications**, all rights reserved. *Java Pro* is an independent publication not affiliated with Sun Microsystems. Sun Microsystems is not responsible in any way for the editorial policy or other contents of the publication. All contents of *Java Pro* are copyright © 1999 by Fawcette Technical Publications, unless otherwise noted. "VBITS" and "Interactive Developer" are trademarks of Fawcette Technical Publications, a California Corporation, James E. Fawcette, President. Java is a trademark of Sun Microsystems. Rather than put a trademark symbol in every occurrence of other trademarked names, we state that we are using the names only in an editorial fashion with no intention of infringement of the trademark. Although all reasonable attempts are made to ensure accuracy, the publisher does not assume any liability for errors or omissions anywhere in the publication. **Canadian GST#: 86831 0764 ISSN: 1096-4495.**

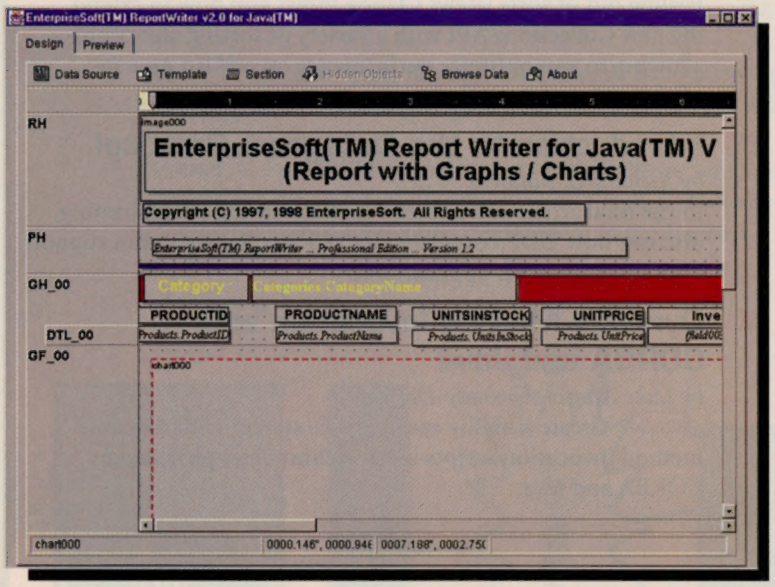
Web These features focus on Java on the Internet.

Java 101 New to Java? These features help get you going.

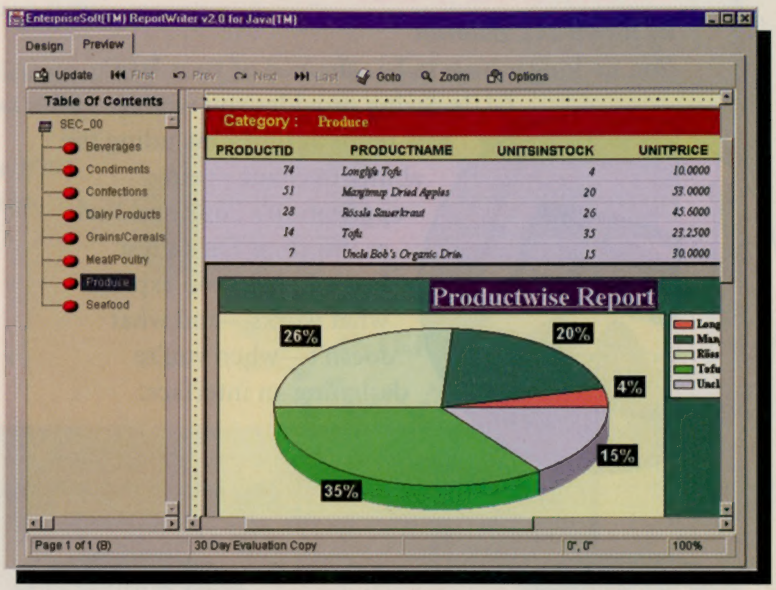
# DO YOU DREAM OF PROVIDING PROFESSIONAL QUALITY OUTPUT FROM YOUR JAVA™ APPLICATIONS/APPLETS?

INTRODUCING

# ENTERPRISESOFT™ REPORT WRITER FOR JAVA™ PRO V 2.0




ERW DESIGNER




ERW VIEWER

## WHAT CAN ERW PRO 2.0 DO FOR YOU ?

**Extract** ERW is data-source independent. It can extract data from any Java application object or from a JDBC data-source. Data extraction is fast and scalable to several thousand objects.

**Analyze** Analyze reports with multiple sections nested groupings, charts/graphs, table of contents for report data, formulas as well as user-defined functions and query operators. The ERW report engine provides blazing fast, real-time report generation.

**Format** Use the familiar, easy-to-use ERW Designer window to format the report structure and use the extensive customization features to define attributes of your report data.

**Output** Any ERW report can be exported to the PDF, HTML or CSV file format. In addition, the reports can be viewed on screen or sent to a printer for a WYSIWYG hard copy.

## THE FIRST REAL REPORTING/PRINTING SOLUTION BUILT WITH YOU, THE JAVA™ DEVELOPER, IN MIND.

- 100% Pure Java™
- Limited, free run-time re-distribution
- Small footprint (~500 KB)
- Robust, fast
- Extendable, scalable
- Easy-to-use designer and API
- Extensive docs and sample code
- Developer-based licensing - $495

**Try ERW Pro 2.0 for yourself!** Find out why companies such as NationsBank/Bank of America, Ford, IBM, and The Getty Museum have chosen ERW Pro 2.0 for their Java™ reporting/printing solutions! Download a 30-day free evaluation copy from our Web site today!

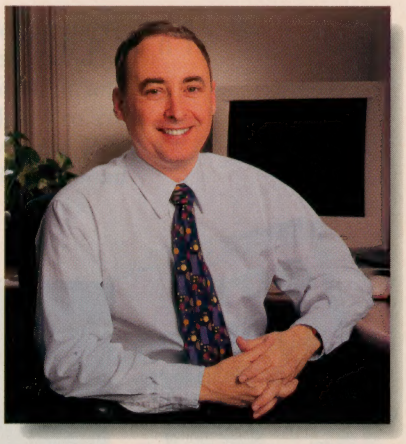

EnterpriseSoft™

7573 Waterford Drive, Cupertino, CA 95014 ■ PHONE 510-742-6700 ■ FAX 510-742-6800
EMAIL java@EnterpriseSoft.com ■ WEB http://www.EnterpriseSoft.com



EnterpriseSoft, the EnterpriseSoft logo, Enterprise Reports and "Enterprise-wide Business Intelligence Solutions" are trademarks of EnterpriseSoft. Java and Java related trademarks are property of Sun Microsystems, Inc. All other trademarks are property of their respective owners.

# Java Grows Up



by Tyler Sperry
*Editor in Chief*

It's a great time to be a Java programmer. Sun has finally shipped JDK 1.2, er, Java 2 and promises any updates will be limited to minor bug fixes. Ditto for Java 1.1, which seems solid with JDK 1.1.7B (at press time). Although most developers are anxious to work with Java 2, it's going to take a while for enterprise Java teams and the vendor community to make the shift. Unlike last year's flurry of different JDK versions, we're now entering a period of sustained interest and relative stability.

Such stability might seem boring, but it's good for business. Last year saw Java openings pay up to 25 percent more than jobs for other languages. Even with funds being diverted to last-ditch Y2K efforts and an impending crop of Java-savvy college students, this year's outlook for Java developers is still good. According to forecasts, analyst's, Java usage in 2000 should be even stronger.

Alas, it is part of the editorial guild rules that an editor should be able to find (or create) storm clouds in a perfectly clear sky. So, in the midst of all these warm Java fuzzies, let me note some anticipated changes.

### You Have Market Share!

Although enterprise managers might find some Java reassurance in the AOL-Netscape merger, I remain skeptical of the alleged Java benefits. AOL needs to stay on the Windows desktop, so Internet Explorer will continue to be AOL's default browser. All credit to the Netscape crew for Gecko, but the fact is Netscape's browser share will continue to slide. This would be more of a tragedy if Netscape hadn't already backpedaled on Java usage last year.

The real story on browsers might come from a revitalized Hot Java effort at Sun. If enterprise developers have a fast, lightweight browsing engine they can incorporate into their designs, we might see a big shift in how they view the Web and protocols like HTTP.

### Open Silliness

Now that Java has graduated from awkward adolescence to an adult programming language, it's time we gave up some of our romantic illusions as well. Yes, it's true: Sun has moved to less-restrictive licensing for Java technology.

In fairness to the folks at Sun, the Java source code was previously available for inspection on their Web site, so it's not like they had been playing the secret API shell game.

Let's give credit to Sun for the change, and to IBM for providing a role model. But don't think for a moment Sun is going to pull a Mozilla number with Java. Eric Raymond and his friends have made some good arguments for the commercial use of Open Source software, but Java isn't Perl or Linux. If you think this is a bad thing, compare Perl and Linux recruitment offers to those for Java developers.

*Let's give credit to Sun for the change and to IBM for providing a role model*

### A Moratorium

As a belated New Year's resolution, I have decided to stop giving Microsoft grief for their Java policies. The company has lost two rounds in the injunction wars—declaring victory each time—and it's become obvious Microsoft intends to treat Java as just another programming language. Accordingly, you might wonder if Visual J++ is destined to be as successful as Microsoft's versions of Lisp, Pascal, and Fortran, but that's not our problem. By producing a proprietary Java stuck at 1.1.4 compatibility, Microsoft is, in effect, betting on the failure of Java 2 in the enterprise. That's a bad bet. But if the gang in Redmond wants to make such a Procrustean bed, we should let them lie in it. Peace. ■

# Become the Master of your domain.

Introducing the Objectsoft family of java tools

**DIFFMASTER**
Works with any file-based Version Control system showing you the differences in your code down to the method level.

**BREWMASTER**
100% Java! Featuring class browser, dynamic syntax parsing, compile management, code query capabilities, Cafe integration, J++ integration and much much more.

**COMMENTMASTER**
Makes your documentation job easier by reading your source code and correcting the comments to conform to the javadoc standard.

**DEBUGMASTER**
Offers a complete debugger environment for JDK or MS JView environments.

[All tools downloadable at our Web Site]

www.objsoft.com

Objectsoft, Inc.

1-888-OBJSOFT

JAVA Pro

Jeff Miller, *Vice President, Publishing*

**Editorial**
Tyler Sperry, *Editor in Chief*
John Dorsey, *Managing Editor*
Elise M. Peterson, *Associate Editor*

**Contributors**
Daniel Savarese, James W. Cooper
Larry O'Brien, Paul Tyma, Rick Grehan

**Technical Reviewers**
Luke Andrew Cassady-Dorion

**Art & Production**
Michael Hollister, *Art & Production Director*
Amy René Pelto, *Design Production Associate*
Christine Marquez, *Production Manager*
Kathy Sweeney, *Production Artist*
James Kattinge, *Senior Designer*
Terrilyn Chance, *Associate Art Director*
Kimberly Bell, Electronic *Products Associate Art Director*
Robyn Wandzell, *Electronic Production Associate*
Kim Le, *Senior Designer*
Michael Furukawa, *Associate Production Manager*

**Advertising Sales**
Kent Wu, *Advertising Director*
Regan Swett, *Regional Sales Manager*
Kelly Keane, *Regional Sales Manager*
Shana Deane, *Regional Sales Manager*
Monica McQueen, *Production Coordinator*
Lilly Gutierrez, *Recruitment Advertising Manager*

**Circulation**
Karen Koenen, *Circulation Director*
Katie Hathaway, *Associate Circulation Manager*
Patrick Bonstead, *Circulation Associate*

**Marketing**
Sherry Thomas-Zon, *Marketing Director*
Julianne Olmstead, *Ancillary Products Manager*
Libby Dickinson, *Marketing Communications Manager*
Margaret Horoszko, *Marketing Designer*
Kristin Fladager, *Marketing Specialist*

**Conferences**
Tena Carter, *General Manager*
Timothy Haight, *Editorial Director*
Ann Odonnell, *Conference Manager*
Daryl Haggard, *Conference Planner*
Janet Brandt, *Conference Planner*
Fred Perry, *Conference Planner*
Robert Scoble, *Conference Planner*
Jennifer Brucker, *Conference Planner*
Tom Donoghue, *Conference Sales Director*
Damian Zusi, *Conference Sales Manager*

**Electronic Products**
Matthew Carter, *Editor & Producer*
Andrew King, *Associate Producer*
Bill Poulin, *Web Developer*
Diane Matcheck, *Product Reviews Editor*
Kay Keppler, *Senior Editor*
Sylvia Zele, *Associate Editor*
Mana Tominaga, *Editorial Assistant*

**Electronic Products, Advertising**
Ferris Ferdon, *Advertising Director*
Ben Waltzer, *Sales Support Manager*
Amanda Tyler, *East Coast Regional Manager*
Brendy Bell, *West Coast Regional Manager*
Jane Myer, *Regional Sales Manager*

**Customer Service**
Andrew Nickels, *Customer Service Manager*
Michelle Grossman, *Customer Service Representative*
Chris Ortiz, *Customer Service Representative*
Melissa Ramirez, *Customer Service Representative*

**Operations**
Trisha Merriam, *Assistant to the Publisher*
Wendy Hartman-Carr, *Controller*
Aaron Wuele, *IS Manager*
Ha Hoang, *IS Specialist*
Raylene Leavitt, *HR/Executive Administrator*
Sam Azzouni, *A/R Clerk*
Lorenzo Valencia, *Facilities Manager*
Daniel Hernandez, *Receptionist*
Ashley Stringfellow, *Office Assistant*

**Fawcette Technical Publications**
James E. Fawcette, *President*
Lynne Matthes, *Group Publisher*
Stephen Evanczuk, *Vice President, Electronic Products*
Chris Kinsman, *Vice President, Technology*
Jeff Miller, *Vice President, Publishing*
Wilson, Sonsini, Goodrich & Rosati, *Corporate Counsel*

# Letters to the Editor

*Letters to* Java Pro *are welcome. To be considered for publication, letters must include your name, address, and daytime phone number. Letters may be edited for form, fit, and style. Please send them to Letters to the Editor,* Java Pro, *c/o Fawcette Technical Publications, 209 Hamilton Avenue, Palo Alto, CA 94301-2500; fax them to 650-853-0230; or e-mail them to java-pro@fawcette.com.*

## Programming Lethargy

I recently received a survey from your company that I would be more than happy to fill out were it not for the fact that it is for Windows only. I have no access to a Windows machine.

Since according to Sun, Java is "a simple, object-oriented, distributed, interpreted, robust, secure, ARCHITECTURE NEUTRAL, PORTABLE, high-performance, multi-threaded, and dynamic language" I am amazed that you decided to send out a survey that is neither ARCHITECTURE NEUTRAL nor PORTABLE. This isn't a simple matter of ignoring your readership—it's laziness.

If you are hoping that Java becomes a stronger and more widely used language, you had better hope that not all developers share your laziness. Because you seem to think developing something as small as a survey needs to be Windows only, why would any developer move from C or C++ to Java? For you and for Java, I hope there is a better reason for your decision.

If I see that your magazine (and this assumes that you begin sending me your magazine) continues to ignore basic components of Java, I would quickly find myself discontinuing my subscription.

*Benjamin Bakken*

*Editors respond: We apologize for any misunderstanding. We hired a research company to take a survey of our readers. That company followed the standard engineering practice of not rewriting code that works, and sent out our survey as part of a Visual Basic application. Although we were struck by the irony of the situation, we also recognized that the vast majority of our readers either use Windows machines for development or know someone who does. We didn't anticipate that this platform dependence would be a problem.*

*We trust the magazine contents will speak to our commitment on cross-platform issues. If readers would like to offer opinions on the magazine with minimal Windows contact, you can send us feedback using the Reader Survey form on page 61.*

*Why would any developer move from C or C++ to Java?*

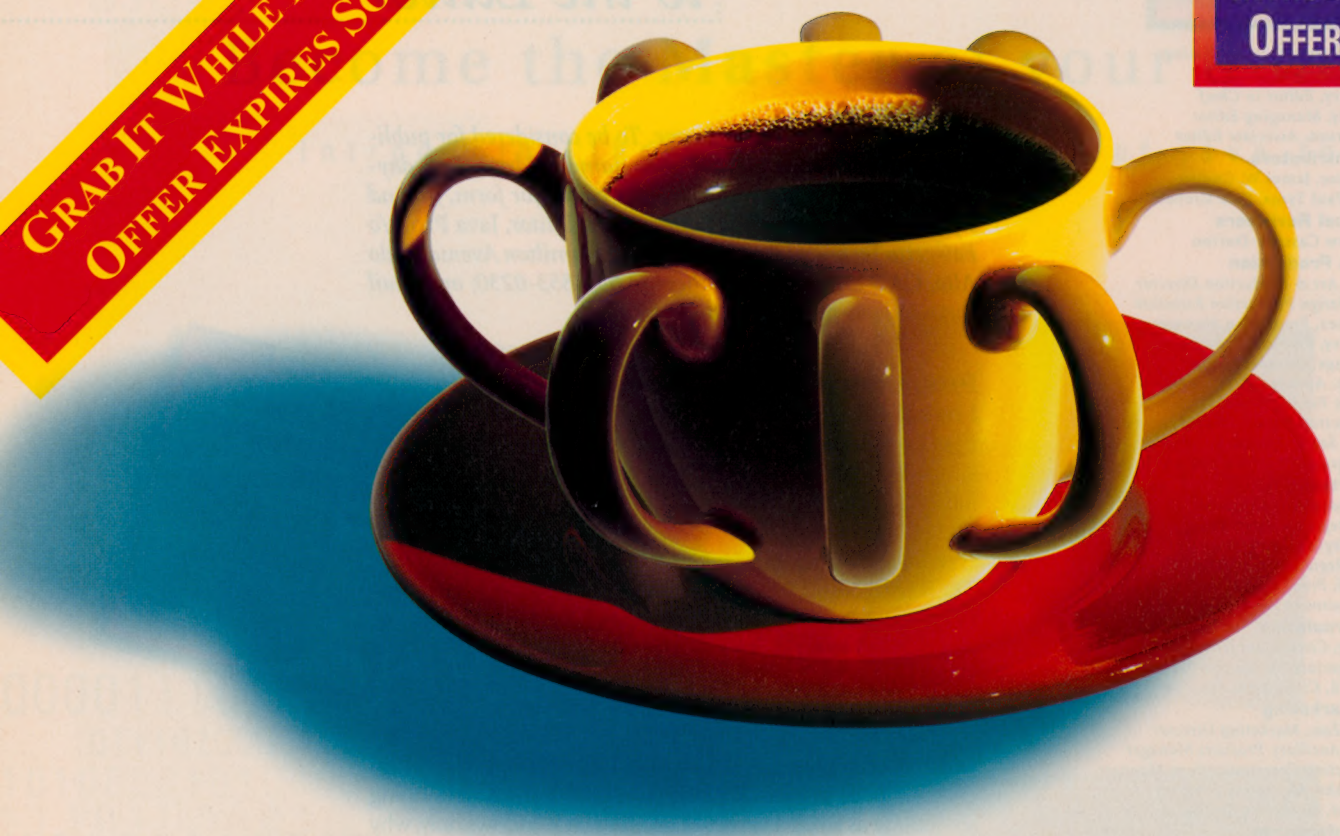
## JBuilder 2 User Responds

I read the review of JBuilder 2 (October 1998) with interest because it is the IDE that my company is using. I would however like to make a few comments about it. It is *very* much a memory hog. My platform is an NT4 workstation with 96 MB of RAM, and it is still slow and locks up at least once a day. I have had the chance to try it on a P400 with 256 MB and it ran fine.

Also, after a couple of months of use, I no longer really use it to design forms visually because I don't like the extra

GRAB IT WHILE IT'S HOT
OFFER EXPIRES SOON!!

LIMITED TIME
INTRODUCTORY
OFFER $995



# IS THIS YOUR JAVA TOOL'S IDEA OF SUPPORTING TEAM DEVELOPMENT?

**Then you obviously don't use PARTS® for JAVA™ Professional, a tool set designed to fully support teams building distributed applications.**

PARTS was created by developers with over 10 years experience designing object-oriented development tools for team programming. Our experience will make a difference in your team's productivity.

PARTS will coordinate your team's efforts with integrated support for PVCS, making it simple to track changes, compare versions, and manage your development projects. Plus, our unique tools help you understand how your code interacts with other pieces of the system, enabling you to clearly see the impact of changes.

PARTS has everything you need to be productive:

- Visual programming—just drag, drop and deploy
- JDBC support for fast, easy database applications
- JFC/Swing for powerful interfaces
- CORBA for distributed applications
- JavaBeans for quick application assembly
- OracleLite, OrbixWeb 3.0, PVCS, jKit/Grid— All integrated for a combined value of $1700!

And all backed by our world class training and consulting program chosen by both Oracle® and Sun®.

Call now for more information on how PARTS will make your team more productive—
1.800.973.4777

**PARTS® FOR JAVA™**
PROFESSIONAL

**BE PRODUCTIVE.**

ObjectShare®

16811 Hale Ave., Suite A, Irvine, CA 92606 www.objectshare.com

PARTS is a registered trademark of ObjectShare, Inc. All other referenced products are trademarks and/or registered trademarks of their respective manufacturers.

## HOW TO REACH US

**Editorial Offices**
Fawcette Technical Publications
209 Hamilton Avenue
Palo Alto, CA 94301-2500
Phone: 650-833-7100, Fax: 650-853-0230
Web: www.java-pro.com, E-mail: java-pro@fawcette.com

**Article Submission**
To submit articles for publication please contact:
Tyler Sperry, Editor in Chief, *Java Pro*
E-mail: tsperry@fawcette.com
Download the Author Guidelines at www.java-pro.com/authguid.htm.

**Product Reviews**
To submit products for review please contact:
Diane Matcheck, Product Reviews Editor
E-mail: dmatcheck@fawcette.com

**Product Announcements**
Please e-mail press releases to: pressreleases@fawcette.com

**Reprints and Permissions**
For all *Java Pro* editorial and advertising reprints contact:
PARS International Corp.
114 East 32nd Street, Suite 503
New York, NY 10016
Phone: 212-779-4469, Fax: 212-779-4277
To quote from an article, please contact:
Monica McQueen
209 Hamilton Avenue
Palo Alto, CA 94301-2500
Phone: 650-833-7147, E-mail: mmcqueen@fawcette.com
Specify the issue date and title of the article, the portion you would like to quote, and the purpose.
**Photocopy Rights:**
Permission to photocopy for internal or personal use may be granted by Fawcette Technical Publications. Please contact Monica McQueen for more information.
**Customer Service and Subscription Information**
For subscription orders, inquiries, or address changes please contact Customer Service:
*Java Pro*
P.O. Box 54584
Boulder, CO 80322-4584
Phone: 303-945-5282, Fax: 650-321-3818
Foreign and Canadian orders must be payable in U.S. dollars, plus postage. The surface rate to Canada and Mexico is $52.97 per year and for all other countries the air mail rate is $78.97 per year.
**Back Issues:**
To order *Java Pro* back issues, call 650-833-7100 or 800-848-5523 and ask for Customer Service. Back issues cost $10. Additional postage will be charged to deliveries outside the USA.

## ONLINE RESOURCES

**Registered Users**
Locator+ code: **JP0299**
Type this code in the Locator+ field at the top of the *Java Pro* home page. You can find all the code in the current issue of *Java Pro* at www.java-pro.com. All the listings and associated files discussed in the articles are available free to Registered members in one ZIP file.

**Premier Club Members**
Members of the Premier Club of The Development Exchange (www.devx.com, $20 for six months) can get each article's listings and associated files in a separate ZIP file, plus archives of all code ever published in *Java Pro*. For example, to download the individual code file associated with "Ask the Pro," by Daniel Savarese, you would type in JP0299AP as the Locator+ code.

**Java Zone**
The Java Zone (www.java-zone.com) on the Development Exchange (www.devx.com) offers additional Java resources including product reviews, programming tips, weekly downloads, and discussion forums.

# Letters to the Editor

*Continued from page 7.*

code it generates and I am starting to use a lot of subclassed Swing components. I also come from a Visual Basic background and was used to its wonderful debugger. I don't use the JBuilder debugger because is slows the execution down even more than normal, and to me is generally pretty useless. Other than that, it's not bad!

*Robert McIntosh*

### Access to Back Issues and Code?

I am looking for the article by Gary Cornell, *Build Your First App(let)* (Winter 1997/1998). Is this article still available on your Web site?

*Clarence Berry*

I think you have a great magazine, and I enjoy buying it each month from McGills bookstore in Melbourne Australia. However, I have signed up as a registered member of the Development Exchange (DevX) and tried to download sample code using the Locator+ code but each time I'm told that I don't have permission to access the code because it's part of the Premier Club, yet in the magazine it quite clearly says at the end of each article the code is free to Registered members of DevX. I don't mind paying for the magazine but I don't think I should have to join the Premier Club just to get code referred to in the magazine, especially since sometime whole listings are referred to but don't appear in some articles. Am I doing something wrong or am I forced to pay twice (once for the magazine and once for Premier Club membership) in order to get the sample code?

*Brett Miller*

*You have a great magazine, and I enjoy buying it each month*

*Editors respond: The complete text of all the articles and columns is available to Premier Club members of the Development Exchange (www.devx.com). Premier Club members can also download the source code for a specific article by clicking on the code link in the related article.*

*Readers who are members at the Registered (free) level can also access the source code for past articles by downloading the issue archive file for the related month using the appropriate Locator+ codes. For example, to download all of the code for this issue you would type JP0299.*

### UML meets Java

Awesome article (*Design Java Apps with UML*, June/July 1998). I just went through the Rational Rose tutorial (included in that month's code archive) and read some other stuff on UML, but other than that I came in green. I am attempting to use UML and Java for my next project, an e-mail client. This gives me a great start. Thanks.

*Mark Sears*

*Editors respond: The Unified Modeling Language (UML) is becoming increasingly popular with Java developers. In addition to using it in our articles to explain things, we anticipate more readers will be using it on a daily basis. To that end, look for an article on Java-oriented UML tools in an upcoming issue.*

# Reviews

DashO-Pro 1.1
jtest! 2.04
JDesignerPro 3.0

## DashO-Pro 1.1

### *DashO Establishes Itself as a Critical Tool for Serious Java Developers*

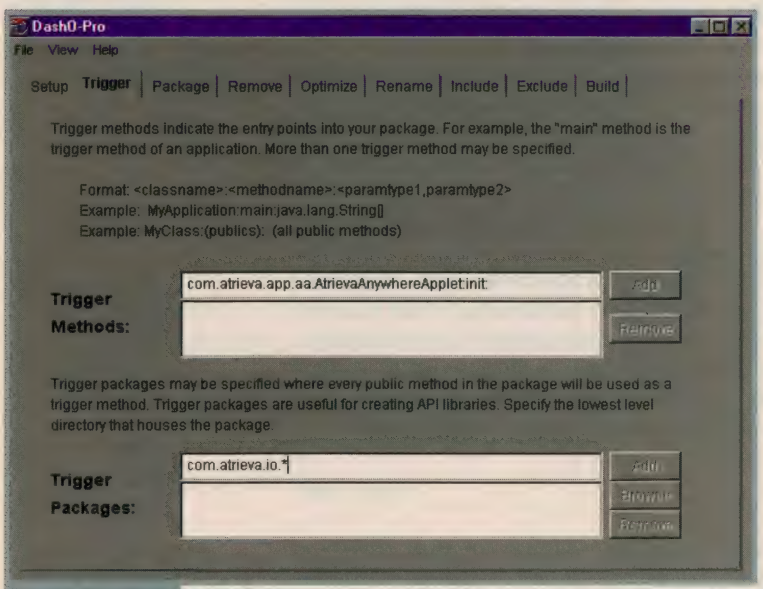
**by Claude Duguay and Susan Schudie**

*Quick Facts:*

*If you need to deliver thin applications over low bandwidth connections, want to protect your code, and make it smaller and faster, DashO is a great investment.*

**preEmptive Solutions Inc.**
26250 Euclid Ave., Ste. 503
Cleveland, OH 44132
**Phone:** 216-732-5895
**Fax:** 216-732-5897
**Web:** www.preemptive.com
**E-mail:** information@preemptive.com
**Price:** $1,145

If you develop Java software and distribute it over the Internet, you already understand the benefits of packaging it in JAR and CAB files. The compression these formats provide reduces download time, creating a more favorable impression with the end user. Making distribution packages even smaller is one of DashO's great strengths. And, if you need to protect intellectual property, DashO provides bytecode obfuscation, which makes reverse engineering attacks unlikely to succeed. For speed-sensitive developers, DashO performs code optimizations that make your programs run faster and more efficiently. Each of these features should be enough to warrant attention, but taken together, DashO establishes itself as a critical tool for serious Java developers.


DashO-Pro
File View Help
Setup Trigger Package Remove Optimize Rename Include Exclude Build
Trigger methods indicate the entry points into your package. For example, the "main" method is the trigger method of an application. More than one trigger method may be specified.
Format: <classname>:<methodname>:<paramtype1,paramtype2>
Example: MyApplication:main:java.lang.String[]
Example: MyClass:(publics): (all public methods)
Trigger Methods:
com.atrieva.app.aa.AtrievaAnywhereApplet:init:
Add
Remove
Trigger packages may be specified where every public method in the package will be used as a trigger method. Trigger packages are useful for creating API libraries. Specify the lowest level directory that houses the package.
Trigger Packages:
com.atrieva.io.*
Add
Browse
Remove

**DashO-Pro GUI** DashO-Pro provides a UI with several pages that lead you through constructing a configuration file for your project. The UI still has some rough edges, such as these less-than-intuitive tab markers.

Billing itself as a "Java Application Packaging Subsystem," DashO operates primarily on classes in a specified path. It applies standard bytecode optimizations, such as devirtualization and desynchronization, and it also leverages other optimizations including renaming and unused code removal. But it stays away from attempting optimizations that a Just In Time (JIT) compiler would typically perform. Code obfuscation, or renaming, is a common way to protect your code from prying eyes by making the bytecode virtually unreadable and reduces overall size by using shorter field, method, and class names.

We found the software installation and upgrades to be less than user friendly. The command line installer led to a few problems during upgrade, which forced us to repeat the process several times. To apply DashO to your code, you need to write a configuration file or use the wizard-style UI to produce the configuration file for you. However, we had limited success with DashO's GUI, which was a bit buggy, exhibiting tab order problems and occasional keyboard problems. Ironically, the UI design itself was adequate, leading the user through the configuration file creation process. But the Swing applet we tried optimizing complained about the main class not being deriving from Applet (Swing uses JApplet) and failed to produce the required configuration file, instead exiting with a rather inelegant exception.

The users manual is concise and complete, although online help would have been useful. The command line options, similar to UNIX commands, are also useful, providing additional control over the optimizations. The error messages were typically instructive enough to lead us to relevant solutions. The only trouble we encountered was related to a few unclear messages that occured when we failed to exclude certain JDK classes, but preEmptive's customer service was very responsive and the problem easily resolved.

Using a configuration file makes it easy to call the program via command line and integrate the optimization as a post-compile step in a build script. To use the configuration files effectively, you have to take the time to work out the kinks. The optimization process can also consume a fair amount of time. preEmptive Solutions reported that it took an hour to optimize the DashO code itself under version 1.1, but only 8 minutes for 1.2 version, currently in beta.

Once we got it working with our configuration file, DashO did a good job of optimizing our code. The compression numbers posted on preEmptive's Web site are consistent with our experience, we averaged 23% and peaked at 30%. The installation and configuration process are not as easy as we would have liked. The GUI solution did not lead us to success and proved to be a source of frustration that clearly needs to be addressed. If you need to deliver thin applications over low bandwidth connections, DashO is a great investment. We expect preEmptive will recognize the need to simplify the installation and will undoubtedly improve the functionality of their wizard in future versions. ■

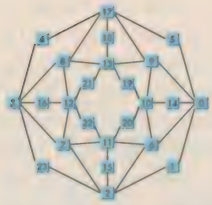
**Claude Duguay and Susan Schudie** work at Atrieva Corp., which provides online backup and retrieval services on the Internet. They have both worked extensively with Java, using it commerically in their development environment. Susan leads development on Atrieva Anywhere, the browser-based part of the Atrieva solution. Claude manages the software engineering team and writes about technology for various magazines.

# TAKE THE BRAKES OFF YOUR JAVA GUI

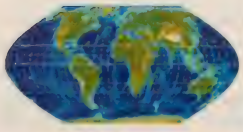
**Drive ILOG JViews 2.0 and get your applications out of the slow lane.** With new ILOG JViews 2.0, you won't be limited to the simplistic forms-based interface that's typical for Java apps. You'll speed into a whole new world of highly interactive, visually rich GUIs that you can build quickly and easily. Your application can now include intelligent, web-based maps, relationship graphs, and detailed, data-aware icons. And like the award-winning ILOG JViews, version 2.0 delivers industry-leading speed and performance.

For new applications or web-enabled versions of existing ones, nothing comes close to ILOG JViews 2.0. **Take a free test drive. Visit our website at www.ilog.com/jv.**

*Create super animated icons faster than ever before*



*Visualize complex data relationships*



*Add "smart" web-based mapping*

*ILOG*
*Tel: 1-800-FOR-ILOG*
*jviews@ilog.com*
*www.ilog.com/jv*

ILOG

*The world leader in software components*

ILOG is a registered trademark of ILOG. ILOG JViews is a trademark of ILOG. All trademarks, service marks and company names are the property of their respective owners.

# jtest! 2.04

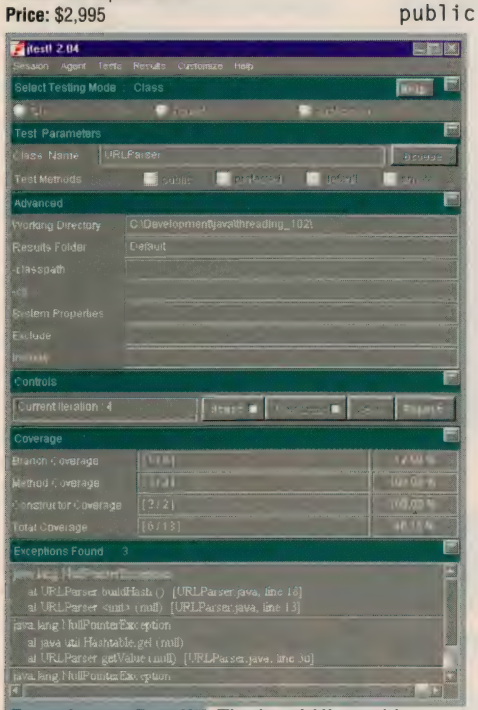
## *Automate White Box Testing*

by Luke Andrew Cassady-Dorion

**Quick Facts:**

*jtest! fully automates white box testing with an intuitive UI and an impressive set of features.*

**ParaSoft Corp.**
2031 S. Myrtle Ave.
Monrovia, CA 91016
**Phone:** 888-304-0041
**Fax:** 626-305-3036
**Web:** www.parasoft.com/products/jtest/
**E-mail:** info@parasoft.com
**Price:** $2,995




**Focusing on Detail** The jtest! UI provides detailed information about the class, applet, or application tested. The results are separated into six sections.

Application testing, an unarguably vital step in the development process, should occur in two distinct forms, black box and white box. In a black box environment, an application's internal structure is treated as an unknown quantity, and its features are probed by the user who has zero knowledge of the decisions made by developers. White box testing attacks the source code from the perspective of the developer. Individual methods are invoked with parameters representing all phases of valid input. For example, consider this method:

```
public int divideThis(int a,
    int b) {
    return a/b;
}
```

Because division will fail when the denominator equals zero, a proper white box testing case for the `divideThis()` method, would invoke the method passing in a non-zero and a zero value for the "b" parameter.

jtest! from ParaSoft not only allows white box testing of classes, it fully automates the process through an intuitive UI and an impressive set of features. The application sequentially invokes methods exposed by classes, continuously checking for uncaught runtime exceptions.

Returning to the earlier division example, jtest! would flag the `divideThis()` method as a potential problem spot. Although pointing out the situation is a benefit to developers, jtest! ignores any potential precondition assumptions you make about various methods. For example, if all code that invokes the `divideThis()` method checks that the denominator is non-zero before invoking the method, there is no reason to perform the check inside the method itself.

jtest! has the ability to execute its checking algorithms against a single class, a collection of classes, an applet, or an application. The decision about when to utilize the tool is one that developers will have to make based on their workflow. In practice, I have found that when working in a group environment it is best to run such checks before checking source code back into a version-control system. Introducing bugs into a system can have far reaching effects if they propagate to other developer's machines.

The figure shows the main jtest! UI as it appears after finding some potential problems in a class file. The lower section of the screen shows each of the methods and information obtained from the stack trace. If the source code is available for the target class, clicking on any of the exceptions identifies the line of code where it occurred. Moving further up the screen, note the section titled "Coverage." Code coverage is an important metric indicating the percentage of your actual code that was executed by jtest! Through execution of methods with all forms of different parameters, jtest! will likely execute as much code as is possible; however, some code is impossible to reach. Consider this code:

```
if(true) {
  System.out.println("this
  will always occur");
}
else {
  System.out.prinlnt("you
  have a better chance of
  winning the lottery and
  getting hit by lightning
  at the same time");
}
```

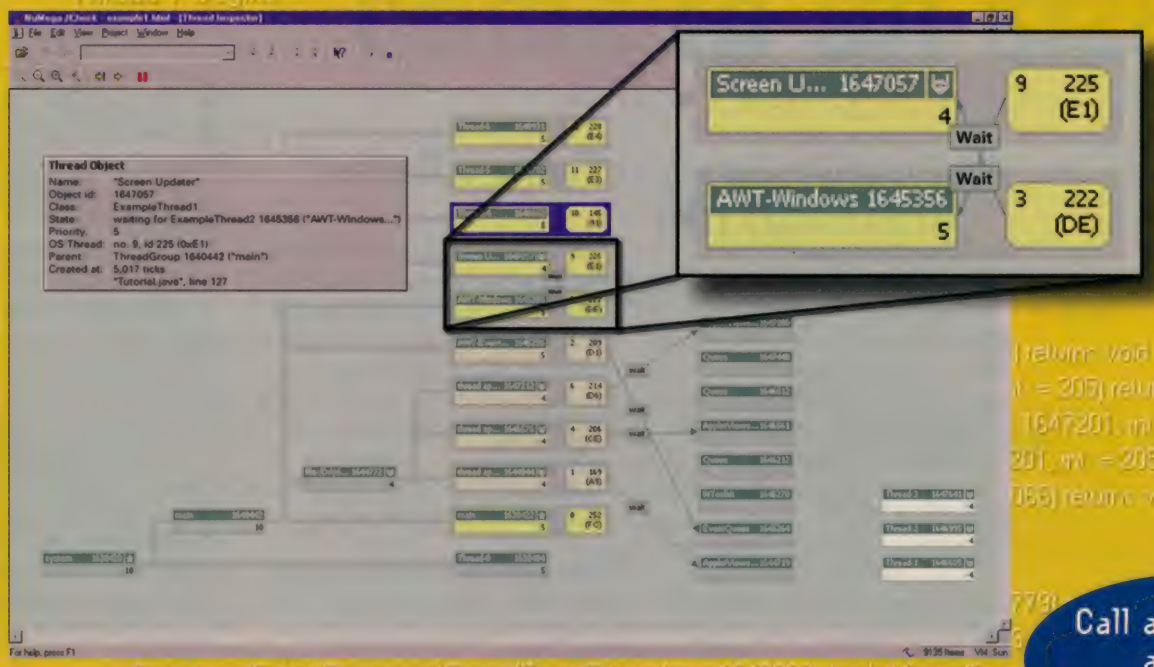
Because the conditional `if(true)` will always execute, the other code will not be reached; this situation would be reflected in the coverage section.

Moving further up the jtest! screen, the section titled "Advanced" allows the configuration of various runtime parameters, both those accepted by the Java VM and ones such as include and exclude file accepted by jtest! Above the "Advanced" section, is a set of configuration options that allows the user to specify which of the various types of methods are to be tested. Often you will only want to test public and protected methods because as your internal private methods might only generate data that does not raise exceptions.

jtest! is a tool I want to add to my arsenal. With ever-shrinking development schedules, any assistance is much needed, and the potential savings provided by jtest! is rather impressive. ■

**Luke Cassady-Dorion,** a Sun Certified Java 1.1 Programmer, is a lead developer with Healtheon Corporation. Luke has deployed many enterprise-wide solutions written entirely in Java. He is the author of numerous articles on distributed computing and Java and co-author of the recently published *Distributed Objects Unleashed!* Luke can be reached at luke@luke.org.

# DEADLOCK!




Call and request a FREE 14 day license

## JCheck™ Exposes Java™ Thread Problems Instantly

Deadlock. Thread Leak. Starvation. Thrashing... Just a few of the problems you'll never have to chase down manually again – if you use NuMega JCheck™, the visual thread and event analyzer for Java. JCheck builds a thread-based graphical model of your program at runtime and provides rich, detailed information on thread states, events, synchronization objects, exceptions, and errors. JCheck expands your runtime visibility in a new dimension, so you will waste less time debugging, and have more time for productive development.

**Take your Java development to the next level. Get NuMega JCheck today!**

### JCheck is also part of NuMega DevPartner™ for Java
### Essential tools for Java development

**NuMega JCheck™** Visual Thread and Event Analyzer

**NuMega TrueTime™** Automatic Performance Analysis and Optimization

**NuMega TrueCoverage™** Automatic Code Coverage Analysis

**1-888-686-3438**

**www.numega.com**

COMPUWARE
People and software for business applications™

30-Day Money Back Guarantee

Copyright © 1998 Compuware Corporation. All rights reserved. All trademarks and registered trademarks are the property of their respective owners.

# JDesignerPro 3.0

## *Application Server is Feature Packed*

**by Luke Andrew Cassady-Dorion**

*Quick Facts:*

*Excellent server combined with robust IDE. The client application is disappointingly slow.*

**Bulletproof Corp.**
15732 Los Gatos Blvd. #525
Los Gatos, CA 95032
**Phone:** 408-374-2323
**Web:** www.bulletproof.com
**Price:** $995 (for the standard configuration, many other configurations at www.bulletproof.com/jdesignerpro/pricing.htm)

Chances are, if you were to ask any software engineer to list the main cause of his stress, he would start with tight development schedules and excessive feature lists. There is a growing demand for applications to be released faster, run faster, and perform every conceivable function. The developer community has responded by creating new tools for rapid application development.

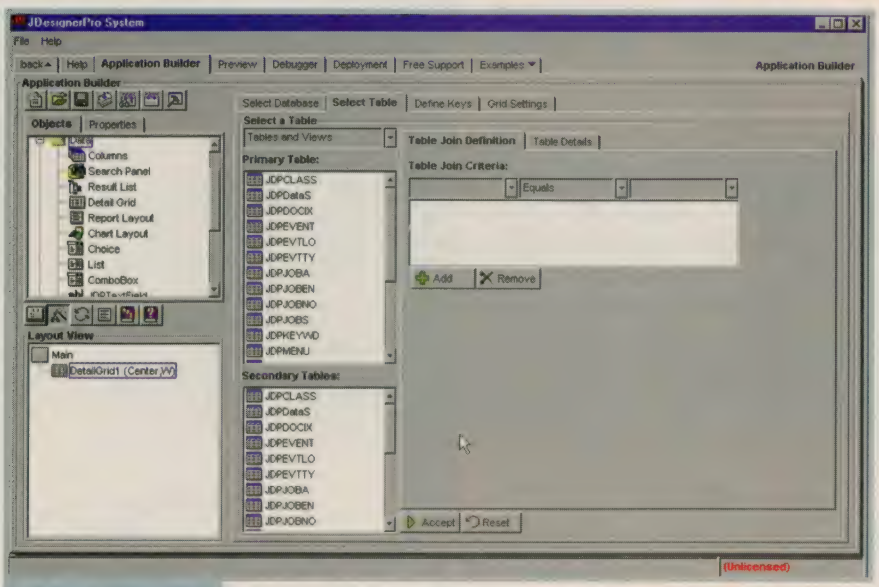
The development of "application servers" is an area of growing interest. The exact definition of an application server varies, but all application servers include a collection of horizontal services that can be leveraged into existing development efforts. What this all boils down to is that instead of writing your own code for functions such as caching, database access, and task scheduling, you use existing code through a published API.

JDesignerPro 3.0, from Bulletproof Corporation, is the newest version of one of the first Java application servers. The environment combines all of the features of a production-grade application server with many features found in today's IDEs to provide an advanced tool for rapid application development and deployment.

### JDesignerPro Architecture

The robust JAGGServer and a client-side tool used in the application development process comprise the JDesignerPro environment. The server can run locally or on a remote server and is active both during development and deployment. The client application is simply a development tool and is not used during the deployment process. Executables produced by the client are used instead.

The JAGG server packs a feature-filled wallop. Applications to be deployed can be scheduled to run at specific times, connect to databases, execute business rules, and perform any feature expected of a modern application server. In testing, I found that the server executed code quickly and would fit well into most business environments.




**Database Interaction** You can use the the SQL statement builder in JDesignerPro to generate database interaction code.

### JDesignerPro Client

The JDesignerPro client application, although rich in features, suffers from slow response time (likely due to the fact that it is written in Java). However, these features, including a robust screen designer and an SQL statement builder, add up to significant time savings for the user. The screen designer uses Java layout managers to give live feedback on the UI being developed, and the SQL statement builder, shown in the figure, interacts with a database to produce bug-free SQL statements.

Although it is nice to have code generated automatically, it is not fully optimized, which can lead to slow execution time. For example, when the SQL builder generates SQL code, it represents the statement as a series of String objects and merges them using the string concatenation operator. Merging String objects using the string concatenation operator instead of the StringBuffer object is a well-known slow down.

The application building process is based on the wizard metaphor. While creating a new application, a developer is moved through a series of steps, each one adding new functionality to the application in development. For example, you might add a grid control to a page in the first step and then attach the control to a database in the second step. At all points in the development process, actions executed generate actual code that can be viewed, modified, and browsed.

I have mixed feelings about JDesignerPro. Although Bulletproof has designed an excellent tool, the implementation could use some work. Granted, they have little control over the speed of UIs developed in Java, but if it wants to compete with IDEs developed in C++, this must be taken into account. It does, however, have an excellent server and seem to have some very talented people designing their applications. ■

**Luke Cassady-Dorion**, a Sun Certified Java 1.1 Programmer, is a lead developer with Healtheon Corporation. Luke has deployed many enterprise-wide solutions written entirely in Java. He is the author of numerous articles on distributed computing and Java and co-author of the recently published *Distributed Objects Unleashed!* Luke can be reached at luke@luke.org.

# Spreadsheet needs come in all sizes.

## Fortunately, so does our technology.

Designed for use on a server or a client, **Formula One Professional** is the most versatile spreadsheet solution available when you need to deploy powerful, Excel-compatible spreadsheets.

Java developers can use it as a JavaBean in development environments such as Visual Café to implement spreadsheet functionality in their applications.

Webmasters can add live spreadsheets to their HTML pages with as few as one line of code.

End users on any desktop can use it as a standalone application.

It is also ideal for thin-client computing and for use on a server to deliver spreadsheet functionality to numerous users.

Besides Java, Formula One Professional delivers its spreadsheet functionality as an ActiveX component, enabling it to be implemented in Windows applications built with development environments such as Visual Basic.

**Formula One™ Professional**

- ✓ Formula One for Java® 5.5.1
  - ✓ Application, applet, and JavaBean
  - ✓ Read/write Excel files
  - ✓ 100% Pure Java
- ✓ Formula One ActiveX™ 6.0
  - ✓ Read/write Excel files
  - ✓ First Impression ActiveX 6.0 charts
- ✓ Fast calculations
- ✓ Excel formula and function syntax
- ✓ Workbook designer
- ✓ Robust API
- ✓ Database connectivity
- ✓ Small footprint and downloads
- ✓ Server and client side functionality
- ✓ Write spreadsheets to HTML
- ✓ Per user and server/CPU pricing

The next time you need robust spreadsheet functionality, make the same choice businesses such as State Street Bank and Simon & Schuster have already made.

Turn to **Formula One Professional**, industry-leading technology for delivering spreadsheets everywhere!

Tidestone TECHNOLOGIES™

*Technology for delivering SPREADSHEETS EVERYWHERE™*

activeX

JDJ EDITOR'S CHOICE AWARD

***INFORMATION:***
**800-922-9665** • ***FREE FULL-FEATURED TRIALS:*** **www.tidestone.com/ssheets**

Copyright © 1999 Tidestone Technologies, Inc. All rights reserved. Tidestone, the Tidestone logo, Visual Components, Formula One, First Impression, and VisualSpeller are trademarks of Tidestone Technologies, Inc. Java, 100% Pure Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. ActiveX is a trademark of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners. Specifications subject to change without notice.

# New Products


Application Servers

Components

Middleware

Books

Developer Tools

Security


## Application Servers

### Inprise Application Server

The Inprise Application Server enables developers to readily build new, platform-independent applications, taking advantage of open industry standards such as CORBA and Java. It provides an integrated end-to-end solution for developing, integrating, deploying, and managing distributed-object applications that support multiple thin clients, mid-tier business logic, and standards-based database and legacy access. Features include open and extensible architecture based on industry standards such as CORBA, C++, Java, and HTML; and support for the Sun Solaris, HP-UX, IBM AIX, and Microsoft Windows NT platforms. Inprise Appplication Server also includes Sun's Java Web Server, and supports other major Web Servers, including Netscape, Microsoft, and Apache. VisiBroker Integrated Transaction Service (ITS), a flexible distributed-object-based transaction service that offers full support for the Java Transaction Service (JTS) standard, and AppCenter, a distributed applications-level management tool, are also included. Call for pricing.

**Inprise Corp.**
**100 Enterprise Way**
**Scotts Valley, CA 95066**
**Phone: 831-431-1000**
**Web: www.inprise.com**

### Bluestone XMLServer

Bluestone XMLServer is a 100% Pure Java Web application server that provides a fundamental platform for the distribution and deployment of XML (Extensible Markup Language) applications. Bluestone XMLServer supports the Enterprise JavaBeans (EJB) specification, and also offers an XML content translator, generator, and transport scripting vehicle. It enables customers to create and read XML pages for communication between Web and non-Web applications and data objects. Bluestone XMLServer quickly automates the exchange of data and provides information integration across heterogeneous applications. Available in January 1999, Bluestone XMLServer will be priced at $2,995 per CPU.

**Bluestone Software**
**1000 Briggs Rd.**
**Mount Laurel, NJ 08054**
**Phone: 609-727-4600**
**1-888-BLUESTONE**
**Web: www.bluestone.com**
**E-mail: info@bluestone.com**

### jBusiness 4

jBusiness 4 is an information integration solution. Using a component-based approach to application development, jBusiness automates the process of creating business objects that encapsulate corporate data. Novera then adds the distribution and management hooks to create enterprise business objects that support the demanding requirements of distributed enterprise applications. Customers can not only choose their preferred IDE but can also integrate Novera's enterprise business objects with industry-standard Web application servers. The jBusiness product includes a Business Object Constructor, an Object Application Server, and a Management Server. jBusiness 4 is available now with development licenses starting at $3,495.

**Novera Software Inc.**
**Burlington Office Centre**
**25 Corporate Drive**
**Burlington, MA 01803**
**Phone: 781-270-4422**
**888-NOVERA1**
**Web: www.novera.com**

### Dynamo Application Server 3.5

Dynamo Application Server is a Java Web application server that features dynamic load distribution, automatic failover, and session recovery. The product supports assembling server applications from server-side JavaBeans components and Java servlets, allowing an open, modular, and reusable architecture. Call for pricing.

**Art Technology Group**
**101 Huntington Ave.**
**Boston, MA 02199**
**Phone: 617-859-1212**
**Fax: 617-859-1211**
**Web: www.atg.com**
**E-mail: info@atg.com**

### Parlay Application Server

Parlay Application Server is an application server and Java development platform for the Internet that lets users visually develop and deploy client- and server-side Java application programs built on the JavaBean component model. The product enables connections to other distributed Java applications and allows new Java-based applications to be integrated with existing transaction systems, other applications, and many databases. The package includes Visual Configurator, a tool that lets you develop and configure distributed Java applications from Enterprise JavaBeans. Call for pricing.

**Information Builders**
**Two Penn Plaza**
**New York, NY 10121-2898**
**Phone: 212-736-4433**
**800-969-INFO**
**Fax: 212-967-6406**
**Web: www.ibi.com**
**E-mail: askinfo@ibi.com**

### Proton 3.0

Proton is a Web application server written in Java. Proton supports multitier Web applications with HTML for presentation layer, Java objects for business logic, and interfacing to enterprise information servers. Proton now supports large-scale enterprise applications with high-performance features, dynamic load balancing, failover management, automatic restart, remote administration and monitoring. Proton provides complete access to Java objects from HTML pages. User interface developers can use simple HTML tools without worrying about the complexity of the business processing on the application layer. Proton uses 100% Pure Java and will run on any platform with Java 1.1 or higher. Developer Edition $1,000; Enterprise Edition $6,000.

**Pramati Technologies Ltd.**
**402 Everest Block**
**Aditya Enclave, Ameerpet**
**Hyderabad 500038**
**India**
**Phone: 91-40-374-3295**
**Fax: 91-40-374-3296**
**Web: www.pramati.com**
**E-mail: info@pramati.com**

## Components

### ICE Browser and ICE Browser Lite

ICE Browser is a 100% Pure Java component that embeds HTML browsing into applications. Developers can use the product for HTML-based help systems embedded in Java applications or as a complete Web browser which can be incorporated into existing applications as a "browser in a window." ICE Browser Lite uses lightweight components from the Swing component set of JFC for all graphics and rendering. Call for pricing.

**ICEsoft AS**
**Høyteknologisenteret**
**Bergen, Norway N-5008**
**Phone: 47-55-54-39-62**
**Fax: 47-55-32-28-53**
**Web: www.icesoft.no**
**E-mail: sales@icesoft.no**

### Jacada Innovator

Jacada Innovator enables new or existing applications written in RPG or Cobol to be deployed with Java or Visual Basic thin clients, eliminating character-based green screens. Jacada Innovator includes a messaging server that facilitates communications between the generated Java or VB client and RPG or Cobol server code, without relying on the 5250 or 3270 data streams. Jacada Innovator allows organizations with legacy applications on mainframes and AS/400s to adopt new development techniques and architectures. Call for pricing.

**Client/Server Technology Inc.**
**400 Perimeter Center Terrace, Ste. 550**
**Atlanta, GA 30346**
**Phone: 770-352-1300**
**800-773-9574**
**Fax: 770-352-1313**
**Web: www.cst-inc.com**
**E-mail: guisys@cst-inc.com**

## Middleware

### DeJINN

DeJINN combines DeCAF messaging middleware framework with a 145K real-time garbage-collecting Java VM to provide information access. The JMS-based Jini application provides CORBA, DCOM, and RMI interoperability. Call for pricing.

**Etsee Soft Inc.**
**114 West Franklin Ave.**
**Pennington, PA 08534**
**Phone: 609-730-9180**
**Fax: 609-730-9663**
**Web: www.etseesoft.com**

### Jacada ComponentServer

Jacada ComponentServer is designed to extend and modernize legacy applications. The product is a specialized application server that publishes an open API. Enterprises can implement new Web and thin-client solutions that access the business logic and data residing in their AS/400 and mainframe applications, without having to modify or understand the application code. Contact for pricing.

**Client/Server Technology Inc.**
**400 Perimeter Center Terrace, Ste. 550**
**Atlanta, GA 30346**
**Phone: 770-352-1300**
**800-773-9574**
**Fax: 770-352-1313**
**Web: www.cst-inc.com**
**E-mail: guisys@cst-inc.com**

### DeCAF

DeCAF is Java messaging middleware for building distributed systems and legacy connectivity solutions. The product includes CORBA, DCOM, RMI, and Socket interoperability. DeCAF is JMS compliant and comes in a three editions: Developer ($199), Pro ($1,199), and Enterprise ($3,999).

**Etsee Soft Inc.**
**114 West Franklin Ave.**
**Pennington, PA 08534**
**Phone: 609-730-9180**
**Fax: 609-730-9663**
**Web: www.etseesoft.com**

## Books

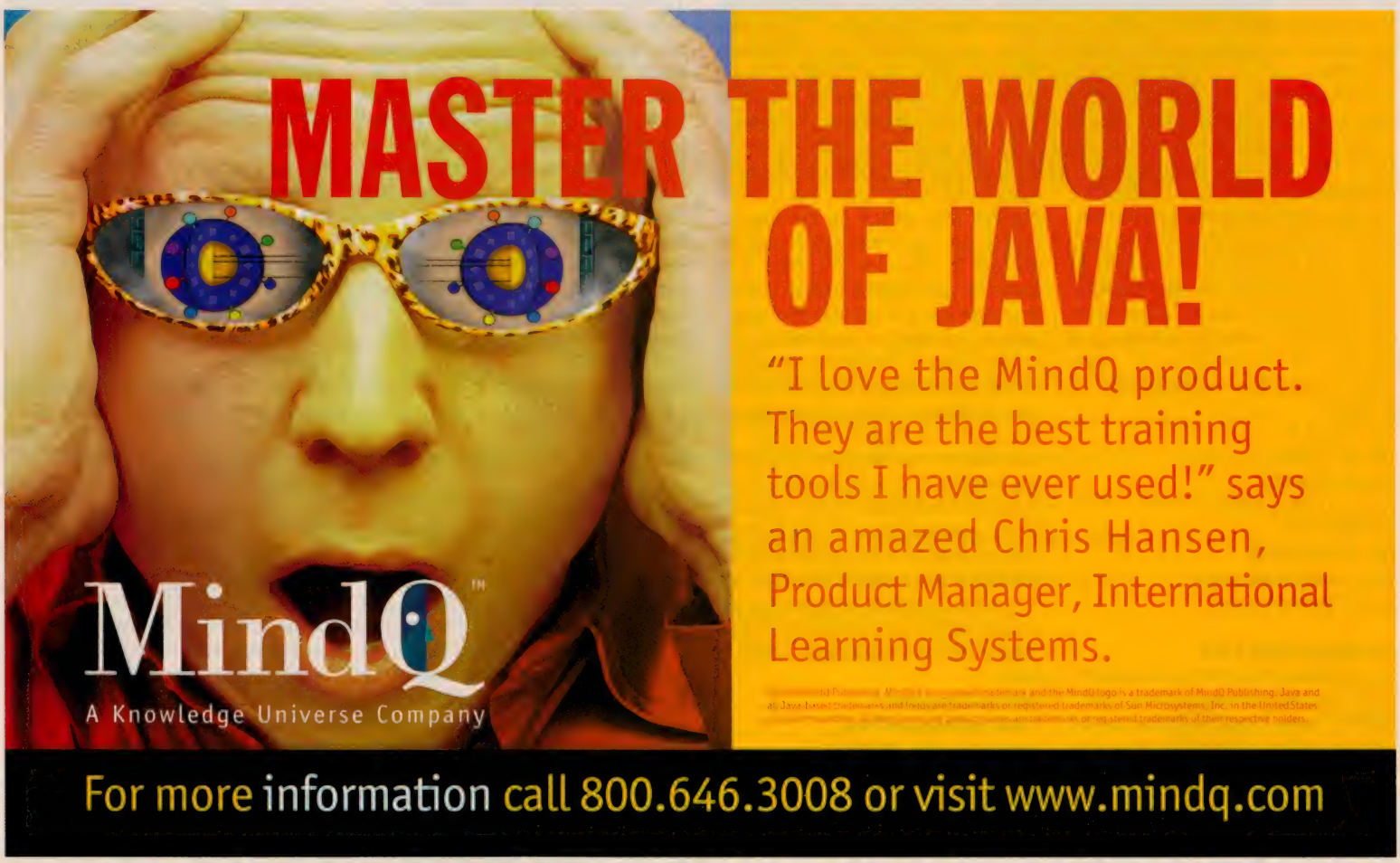
### Java Servlet Programming

*Java Servlet Programming* covers the Java Servlet API, a standard extension to Java that provides a generic mechanism for extending the functionality of any kind of server. It includes over 100 examples you can use as the basis of your own servlets. By Jason Hunter with William Crawford. ISBN:1-56592-391-X, $32.95.

**O'Reilly & Associates Inc.**
**101 Morris St.**
**Sebastopol, CA 95472**
**Phone: 707-829-0515**
**800-998-9938**
**Fax: 707-829-0104**
**Web: www.oreilly.com**

### Java I/O

*Java I/O* tells you what you need to know about streams, which provide simple ways to read and write data of



MASTER THE WORLD OF JAVA!
"I love the MindQ product. They are the best training tools I have ever used!" says an amazed Chris Hansen, Product Manager, International Learning Systems.
MindQ™
A Knowledge Universe Company
For more information call 800.646.3008 or visit www.mindq.com

different types. It shows you how to control number formatting, use characters aside from the standard ASCII character set, and get a head start on writing truly multilingual software. By Elliotte Rusty Harold. ISBN: 1-56592-485-1. $32.95.
**O'Reilly & Associates Inc.**
**101 Morris St.**
**Sebastopol, CA 95472**
**Phone: 707-829-0515**
**800-998-9938**
**Fax: 707-829-0104**
**Web: www.oreilly.com**

## Developer Tools

### JClass 3.6

JClass 3.6, KL Group's JavaBeans suite of products, offers support for Java 2. Developers can use JClass to create professional GUI applications. The JClass family includes six products. JClass HiGrid is a RAD outline-grid for developing advanced multi-level data-bound database GUIs. JClass DataSource is a hierarchical multiple-platform data source that provides the power of data binding to your components. JClass LiveTable is a grid/table for displaying and manipulating tables and forms. JClass Chart is a powerful component to embed sophisticated graphs and charts. JClass BWT is a Java GUI toolkit that enhances and extends AWT with over 20 components. JClass Field is a 100% Pure JavaBean for Java and JavaScript developers that provides data input and validation for a range of popular data types. Two Suites of JClass components are also available: JClass Enterprise Suite and JClass Standard Suite. An evaluation download of JClass 3.6 is available at www.klgroup.com/jclass. Call for pricing.
**KL Group**
**260 King St. E.**
**Toronto, Ontario**
**Canada M5A 1K3**
**Phone: 416-594-1026**
**800-663-4723**
**Fax: 416-594-1919**
**Web: www.klgroup.com**
**E-mail: direct@klg.com**

### StructureBuilder 3.0.2

StructureBuilder is a family of Java component, application, and business object building tools. StructureBuilder can be used as a stand-alone application or in conjunction with Java IDEs and application servers. Developers can specify and display the relationships between objects, graphically develop complex object models, and generate customized code for their applications. The model and the code are synchronized, so any change to the Class diagram is instantly reflected in the code and any change to the code is instantly reflected in the diagram. StructureBuilder Pro adds Sequence diagrams. Create and share both notes and class model information between the different diagrams and click on objects and messages in the sequence diagram to view the corresponding code. StructureBuilder Pro-SM adds Sequence Methods, a new technology that allows a programmer to develop code from sequence diagrams. Prices start at $695.
**Tendril Software**
**319 Littleton Rd. #105**
**Westford MA 01886**
**Phone: 978-392-9600**
**Fax: 978-392-9877**
**Web: www.tendril.com**

### NuMega DevPartner 1.2 for Java

NuMega DevPartner 1.2 for Java is a suite of developer productivity tools focused on improving debugging, tuning, and testing applications on multiple Java virtual machines. DevPartner 1.2 for Java consists of three individual tools. NuMega JCheck 1.2 is a visual thread and event analyzer. JCheck reduces the amount of time it takes to debug difficult problems in multithreaded Java applications and components by building a graphical model of the Java program that provides detailed information on its threads, events, and synchronization objects. NuMega TrueTime 1.2 Java Edition is an automatic performance analysis and optimization tool that enables Java developers to optimize the speed and usability of their software. TrueTime automatically locates performance bottlenecks and reports application and component performance data. NuMega TrueCoverage 1.2 Java Edition is an automatic code coverage analysis tool. TrueCoverage allows developers and testers to automatically locate untested code in Java applications and components. TrueCoverage helps development teams save testing time and improve code reliability by measuring and tracking code execution and code-base stability during development. Priced at $599.
**Numega Compuware**
**31440 Northwestern Hwy.**
**Farmington Hills, MI 48334**
**Phone: 603-578-8400**
**800-468-6342**
**Fax: 603-578-8401**
**Web: www.numega.com**
**E-mail: info@numega.com**

### Jikes Open Source

Jikes is a Java compiler that translates Java source files as defined in *The Java Language Specification* (Addison-Wesley, 1996) into the bytecoded instruction set and binary format defined in *The Java Virtual Machine Specification* (Addison-Wesley, 1996). Jikes accepts the Java language only as specified: not as a subset, variant, or superset. In addition to strictly adhering to specifications, Jikes is faster than most compilers and can compute the complete dependency relations in program files. This allows the generation of dependency makefiles suitable for use with make. Until now Jikes has only been available in binary form. IBM Research is now making the source code available as open source. See the Jikes Web site at www.ibm.com/research/jikes for additional information.
**IBM Corp.**
**Web: www.alphaworks.ibm.com/formula/Jikes**
**E-mail: alphawks@us.ibm.com**

### Virtual Instrumentation Beans 2.0

Virtual Instrumentation Beans (VIB) is a suite of JavaBeans used to create Java-based graphical user interfaces (HMI, MMI, SCADA, and so on) for factory, laboratory, home, and facility automation applications. The new 2.0 Professional release includes support for RMI network and optional support for CORBA servers. The single developer version has also been enhanced for this release. VIB 2.0 Professional has been certified as 100% Pure Java by Sun Microsystems. Developer Edition $399; Pro $1,299.
**ErgoTech Systems Inc.**
**P.O. Box 1287**
**190 Central Park Square**
**Los Alamos, NM 87544**
**Phone: 505-662-5156**
**888-ERGOTEC**
**Fax: 505-662-7442**
**Web: www.ergotech.com**
**E-mail: javabeans@ergotech.com**

### Java/Perl Lingo

Java/Perl Lingo (JPL), software that enables programmers to use the strengths of both Java and Perl in the same environment, is now available as open source software. JPL enables programmers to implement Java methods with Perl and to use Perl code to access Java via the Java Native Interface (JNI). It includes a translator and build system that makes it easy to create JPL applications. The JPL tool and its source code are available free as part of the latest development release of Perl (version 5.005_54) and can be obtained at www.perl.com/.
**O'Reilly & Associates Inc.**
**101 Morris St.**
**Sebastopol, CA 95472**
**Phone: 707-829-0515**
**800-998-9938**
**Fax: 707-829-0104**
**Web: www.perl.com**

## Security

### BSAFE Crypto-J

RSA BSAFE Crypto-J provides developers with an implementation of privacy, authentication, and data integrity, routines—all in Java. Crypto-J contains a full palette of popular cryptographic algorithms and is a Java Security Package that was designed to be called either through the Java Security API or directly through Crypto-J's API. Written in Java, Crypto-J is compatible with Sun's JDK 1.1 and higher. Crypto-J is fully compliant with the Public-Key Cryptography Standards (PKCS), the industry standards for cryptography. Crypto-J also includes implementations of the government standard DES and SHA1 algorithms. It enables developers to create secure electronic commerce, home banking, webcasting, and enterprise security applications. Call for pricing.
**RSA Data Security**
**2955 Campus Dr., Ste. 400**
**San Mateo, CA 94403-2507**
**Phone: 650-295-7600**
**Fax: 650-295-7700**
**Web: www.rsa.com**
**E-mail: products@rsa.com**

### BSAFE SSL-J

SSL-J supports SSL v3 specification details for interoperable functionality between SSL clients and servers. It provides everything needed to include SSL. Technical assistance is provided through extensive sample code, API libraries, certificate authority components, full user and API documentation, and technical support. SSL-J also offers support for X.509 certificates for server and client authentication, and verification ensures interoperability with Netscape, Microsoft, and VeriSign certificate schemes. Call for pricing.
**RSA Data Security**
**2955 Campus Dr., Ste. 400**
**San Mateo, CA 94403-2507**
**Phone: 650-295-7600**
**Fax: 650-295-7700**
**Web: www.rsa.com**
**E-mail: products@rsa.com**

Microsoft

Where do you want to go today?®

With Visual J++ you can:

Visually design Windows and Web applications

Quickly create server-side business components

Easily debug and deploy multi-tier solutions

## NEW VISUAL J++ 6.0

Finally, a Java language tool for the **PLATFORM** you see most.

Microsoft's Visual J++® 6.0 Professional Edition development system is the only tool for Java optimized for Windows® and the Web. This means you can now create Java applications offering the functionality, performance, and integration with existing software your users demand. Take advantage of Windows Foundation Classes (WFC), an object-oriented framework that simplifies and unifies Windows-based and DHTML programming. Go on to create COM components that integrate with existing business systems. Then instantly deploy applications to any remote file or Web server, making distribution and configuration no big deal. And it's all in a highly-productive, visual RAD environment. To preview this new tool, especially designed for the world's most popular platform, go to msdn.microsoft.com/visualj

Microsoft®
VISUAL
Studio

© 1998 Microsoft Corporation. All rights reserved. Microsoft, Windows, *Where do you want to go today?*, Visual J++, Visual Studio, and the Visual Studio logo are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Works With
JDK 1.1/
Swing 1.0.3,
JDK 1.2 beta 4

# A Test Panel for Pluggable Look & Feel

## Here's a tool you can use to test your GUI against all the L&F implementations installed on your system at run time

by Richard G. Baldwin

It used to be that the overall visual and functional quality of GUIs produced with Java was considered second-rate. The introduction of Swing has changed all of that by providing an alternative to the Abstract Windows Toolkit (AWT) for creating GUIs—you no longer need be ashamed of the quality of your Java GUIs. Swing provides a set of lightweight components that can be used in place of the components in the AWT, enabling you to produce high-quality GUIs, along with many components not contained in the AWT, such as progress bars, tool tips, trees, and combo boxes.

Richard Baldwin is a Professor of CIS/CSC at Austin Community College in Austin, Texas, where he teaches introductory, intermediate, and advanced Java Programming courses. He also provides Java consulting and onsite training for various companies in central Texas. He publishes Java tutorials at www.phrantic.com/scoop/onjava.html. Reach him at baldwin@austin.cc.tx.us.

**Listing 1 Testing JSlider** This is a minimal GUI designed to instantiate and link to a PLAF Panel. In addition, this program places an inactive JSlider in the GUI to illustrate the effect of changing the look and feel.

```
/*
This program was tested using JDK1.1.6
and Swing 1.0.1 under Win95.
*************************************/

import java.awt.event.*;
import java.awt.*;
import com.sun.java.swing.*;
import java.util.*;

public class TestGui01 extends JFrame {
  public static void main(
                        String[] argv){
    new TestGui01();//instantiate obj
  }//end main()
  //---------------------------------//

  TestGui01(){//constructor
    getContentPane().setLayout(
                      new FlowLayout());
    getContentPane().add(
                        new JSlider());


    setSize(250,75);
    setTitle("Baldwin's Test GUI");
    setVisible(true);

    //Instantiate and link to a
    // PlafPanel
    new PlafPanel03(this);
    //=================================//

    //Inner class terminates program
    // when JFrame is closed.
    this.addWindowListener(
                  new WindowAdapter() {
      public void windowClosing(
                        WindowEvent e) {
        System.exit(0);}});//end class
  }//end constructor
}//end class TestGui01
```

This article will focus on one of the improvements that Swing offers—Pluggable Look and Feel (PLAF). Swing makes it possible to change the look and feel (L&F) of a Swing GUI at run time. You can design the components anyway you like and change how they look and feel while the program is running. Swing 1.0.1, which is compatible with JDK 1.1.6, comes with three different PLAF implementations right out of the box: Metal, Motif, and Windows. Similar capability is included in JDK 1.2 beta 4.

If you aren't satisfied with the standard PLAF implementations, you can create your own L&F, but be aware that the programming effort may not be trivial. If you create a GUI using exclusively Swing components, you can provide your customer with the ability to switch between L&F implementations at run time. A sample program will be provided later in this article that demonstrates this capability. This article will concentrate on PLAF and will assume that you already understand how to use the AWT to create and service your GUIs.

Changing the L&F of a GUI can produce unexpected results. For example, Components that are properly spaced in one L&F implementation might appear overcrowded in another. It is important that you test your GUI for different L&F implementations. The PLAF control panel (which I will refer to as a PLAF Panel) allows you to hook into your GUIs when you design them so that you can test them against all of the L&F implementations installed on your machine. Only after testing will you know if a particular GUI layout is suitable for all available L&F implementations. When you have finished L&F testing and have decided which L&F implementation to use, remove the PLAF Panel by eliminating one statement from the constructor for your GUI.

**Editor's Note:** The code for this article was developed using JDK 1.1.6 and tested with JDK 1.2 beta 3. Alas, the package name conventions have changed since the release of those tools. For details, see Sun's explanation at http://java.sun.com/products/jfc/package.html. If you are using tools that were distributed in 1998, then you are probably safe using the import line already in the listings:

```
import com.sun.java.swing.*;
```

Packages using the com.foo.* naming style are proprietary. Now that Swing is considered one of the core packages for JDK 1.2, it has taken up residence in the javax extensions neighborhood. If you plan to develop using the post-beta releases of Swing 1.1 or JDK 1.2, you will need to change most of your import statements to:

```
import.javax.swing;
```

Should you find this arrangement logical, please bear in mind that the release notes for Swing 1.1 mention that the com.sun.java.swing .plaf.motif and com.sun.java.swing.plaf.windows packages have not moved into javax.swing. If you have already begun development with the old package naming, you'll want to check out either Sun's free renaming utility at http://java.sun.com/products/jfc/PackageRenamer/ or WoodenChair Software's namespace manager product at www.woodenchair.com.

## Test Program

Before getting into the complexity of the PLAF Panel program, I want to introduce a simple test GUI program, TestGui01.java, that can be used to exercise the PLAF Panel once it is available (see Listing 1). The test program and the PLAF Panel program were successfully tested using JDK 1.1/Swing 1.0.3 and JDK 1.2 beta 4.

This program illustrates how to hook a PLAF Panel to a GUI and the dramatic effect of changing the L&F on the GUI. I kept the test program simple so that the relationship between the test GUI and the PLAF Panel wouldn't get lost in the clutter. This program instantiates a JFrame object and places a JSlider object in the client area of the frame. No code is provided to support the slider. The objective is to illustrate the visual aspects of the JSlider component under different L&F implementations without getting lost in how to use the slider. You will see that the visual rendering of the JSlider component is significantly different among the different L&F implementations (see Figure 1).

This code shows the beginning of the class definition and the definition of the `main()` method for the test GUI program. The `main()` method instantiates an object of the class:

```
public class TestGui01 extends JFrame {
  public static void main(
                    String[] argv){
    new TestGui01();//instantiate obj
  }//end main()
```

This code shows the constructor for this class with some of the common code omitted and replaced by comments:

```
TestGui01(){//constructor
    getContentPane().setLayout(
                      new FlowLayout());
    getContentPane().add(new JSlider());

    //setSize, setTitle, setVisible

    //Link to a new PlafPanel
```

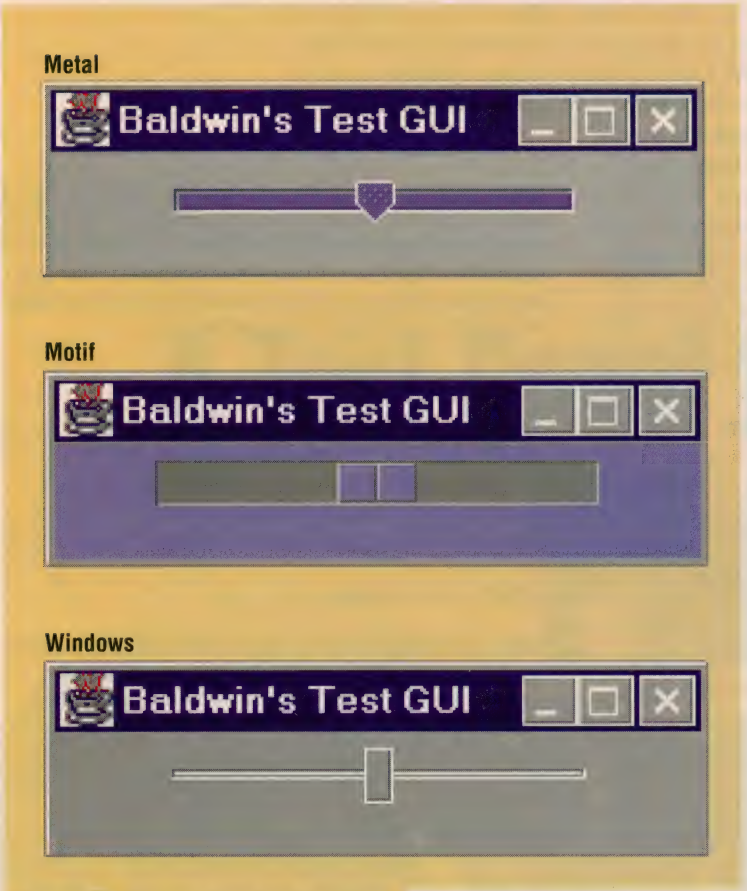



**Figure 1 Choose Your Look and Feel** The Swing classes provide the Metal, Motif, and Windows L&F for your GUIs. With careful testing you can ensure consistent layout from one L&F to another.

```
    new PlafPanel03(this);

    //Anonymous inner class terminates
    // program when JFrame is closed.

  }//end constructor
}//end class TestGui01
```

You might not be familiar with the requirement to use `getContentPane()` for setting the layout and adding the JSlider component to the JFrame object. This is a Swing requirement that does not exist when programming with the AWT (See "Resources" for more information). You should be familiar with `setSize()`, `setTitle()`, and `setVisible()`, and the instantiation of an anonymous inner class to terminate the program when the user presses the close button on the frame.

That leaves only one statement in the program and, for our purposes, it is the most important statement in the entire program:

```
new PlafPanel03(this);
```

This statement instantiates a `PlafPanel03` object, passing a reference to the GUI under test to the constructor for the PLAF Panel. From this point, when one of the buttons is selected on the PLAF Panel to change the L&F, that change will apply both to the GUI under test and the PLAF Panel (see Figure 2). This statement is all that is required to hook a PLAF Panel to your GUI and make it possible to test your GUI for all of the L&F implementations currently installed on your system. This is also the statement that you can disable to remove the PLAF Panel after testing.

## PLAF Panel Program

To hook the PLAF Panel to a user interface under test, instantiate a PLAF Panel object, passing a reference to the JFrame containing your user interface when the PLAF Panel is constructed (see Listing 2). The only link between the two is that the PLAF Panel has a reference to the user interface so that it can change the L&F of the user interface.

The `PlafPanel03` class in Listing 3 extends JFrame. Therefore, an instance of the class is a JFrame. When displayed, the JFrame object contains one JButton for each L&F implementation currently installed in the system. Clicking each JButton will cause the L&F of the user interface under test to change to the L&F represented by that JButton. (The L&F of the PLAF Panel also changes.) The name of each L&F is automatically extracted from its class and displayed on the corresponding JButton.

This code is the declaration of an unusual reference variable type that is used to refer to an array of L&F information:

```
UIManager.LookAndFeelInfo[]
plafInfoArray;
```

This declaration is a little unusual because the ActionEvent class is an inner class of the UIManager class. This explains the period in the middle of the type declaration. The Swing documentation contains the following description of the ActionEvent class:

*"Provide[s] a little information about an installed LookAndFeel for the sake of configuring a menu or for initial application set up."*

With that explanation, the beginning of the class definition and the unusual reference variable declaration follow:

```
public class PlafPanel03 extends JFrame {
  UIManager.LookAndFeelInfo[]
    plafInfoArray;
```

In order for the event handlers in an object of this class to control the L&F of a test GUI, the object must have a reference to a JFrame object containing the test GUI. Such a reference is passed as a parameter to the constructor. Also,

**You no longer need be ashamed of the quality of your Java GUIs**

in order for the event handler to control the L&F of the PLAF Panel, it must have a reference to the PLAF Panel object. This code shows the variables used to save these two required references:

```
JFrame testGui;
PlafPanel03 thisPlafPanel = this;
```

This code shows the beginning of the constructor for the PLAF Panel. The first two statements in the constructor save the incoming reference to the GUI object under test, and create a `JLabel` that contains a descriptive label for the PLAF Panel:

```
public PlafPanel03(JFrame testGui) {
  this.testGui = testGui;
  this.getContentPane().add(
    new JLabel("=========="
    + "Plaf Selection Panel, "
    + "R.G.Baldwin"
    + "=========="));
```

The single statement in this code gets an array containing a list of all of the L&F implementations contained in the current system and stores it in the unusual reference variable discussed earlier:

### Resources

The PLAF Papers: http://java.sun.com/products/jfc/tsc/plaf_report/plaf_report.html

Swing, Understanding getContentPane() and other JFrame Layers, Lesson 087: www.phrantic.com/scoop/onjava.html

# Introducing the first 100% Java™ Internet map server.

UNIX® NT® Whatever.

**MapInfo MapXtreme™**
NT and NEW JAVA™ Edition

**PUT A MAP IN IT**

**DOWNLOAD IT NOW**
**www.mapxtreme.com**

1.888.MAPINFO
www.mapinfo.com

*©1998 MapInfo Corporation. All rights reserved. The MapInfo logo is a registered trademark and MapInfo MapXtreme is a trademark of MapInfo Corporation in the United States. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All other marks, trademarks and registered trademarks are the property of their respective holders.*

03601

MapInfo®
THE INFORMATION DISCOVERY COMPANY

```
plafInfoArray =
  UIManager.getInstalledLookAndFeels();
```

Once the list of L&F implementations is available, the program needs to create a set of JButton objects that will be used to invoke each implementation. Because I didn't know how many L&F implementations might be installed in a system, I didn't know how many buttons would be required. Therefore, I used a Vector object to contain references to those JButton objects. This makes it possible to determine the number of buttons at run time, rather than having to establish the number at compile time. I instantiated the JButton objects in a for loop, which makes one iteration for each element in the array of L&F implementations.

This code shows the declaration of the Vector, the beginning of the for loop, and a statement to populate one element of the Vector:

```
Vector theButtons =
      new Vector(plafInfoArray.length);
  for(int cnt = 0;
           cnt < plafInfoArray.length;
           cnt++){
    theButtons.addElement(new JButton());
```

The program needs the name of the L&F class associated with each of the elements in the L&F array, so I used this code to extract the name and save it in a reference variable named `theClassName`:

```
//Still in for loop
String theClassName =
  plafInfoArray[cnt].getClassName();
```

You can use the L&F classes to create labels for each of the JButton objects used to invoke the new L&F. The String representation of the entire class name is too long to serve as a label on a button, so I extracted a short name for each L&F implementation and used it to label the JButton corresponding to that L&F. The short name appears following the last period in the String representation of the class name for the L&F implementation. I did some simple string manipulation to extract the name. Note the requirement to downcast the reference to the JButton extracted from the Vector. This downcast is necessary because the reference is stored in the Vector as type Object:

```
//Still in the for loop
String label =
  theClassName.substring(
  theClassName.lastIndexOf(".")+1);
((JButton)theButtons.
  elementAt(cnt)).setText(label);
```

Next, I instantiated and registered an ActionListener object on each of the JButton objects. When the button is clicked, the code in the `actionPerformed()` method of the listener object causes the L&F to be changed. As I mentioned previously, because the references to the buttons are stored as Objects in a Vector, it is necessary to downcast them before they can be used. The ActionEvent handler also needs the name of the L&F class, so I passed it as a parameter to the constructor for the listener object:

```
//Still in for loop
((JButton)theButtons.elementAt(cnt)).
  addActionListener(new MyActionListener(
  theClassName));
```

Finally, while still in the for loop, the program adds each JButton to the JFrame, ending the for loop:

```
  //Still in for loop
```

**Figure 2 GUIs to Be Proud Of** After instantiating a PLAFPanel03 object, when one of the buttons is selected on the PLAF Panel to change the L&F, that change will apply both to the GUI under test and the PLAF Panel.

**Listing ② Demonstrating Pluggable Look and Feel** This program creates a GUI in a JFrame containing a PlafPanel. The PlafPanel03 is instantiated from a custom class named PlafPanel03 and contains one JButton for each L&F implementation in the currently installed JDK.

```
import java.awt.*;
import java.awt.event.*;
import com.sun.java.swing.*;
import java.util.*;

// Subclass JFrame to display a window
public class Swing04 extends JFrame {//controlling class

  Swing04(String title) {//constructor
    // Set the title of the JFrame
    super(title);

    //Now create the test GUI for this application.
    getContentPane().setLayout(new FlowLayout());

    //Add some components to be viewed with the new L&F
    getContentPane().add(new JButton("Dummy JButton"));
    getContentPane().add(new JTextField(
      "Dummy JTextField"));

  }//end constructor
  //-----------------------------------------------------//

  // Create main method to execute the application
  public static void main(String args[]) {
    JFrame jFrame = new Swing04(
      "Pluggable L&F, Copyright 1998, RG Baldwin");

    // Set the size of the JFrame and show it
    jFrame.setSize(400,100);
    jFrame.setVisible(true);

    // Create a WindowAdapter to terminate the program
    // when the window is closed.
    jFrame.addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        System.exit(0);
      }//end windowClosing()
    });//end WindowListener

    //Create a PlafPanel in a floating JFrame that can be
    // used to test this GUI.
    JFrame myPlafPanel = new JFrame(
      "PL&F Panel,Copyright 1998,RG Baldwin");
    myPlafPanel.getContentPane().add(
      new PlafPanel03(jFrame));
    myPlafPanel.setBounds(0,400,400,150);
    myPlafPanel.setVisible(true);

  }//end main()
}//end class Swing04
```

**Listing ③ The PlafPanel03 Class** This class constructs an object that can easily be hooked to a GUI to test the GUI for all of the look and feel implementations currently installed on the system. To hook this object to a GUI under test, pass a reference to the JFrame containing the GUI as a parameter when this object is constructed.

```
/**************************************
Tested using JDK1.1.6 and Swing 1.0.1
under Win95.
**************************************/

import java.awt.event.*;
import java.awt.*;
import com.sun.java.swing.*;
import java.util.*;

public class PlafPanel03 extends
                            JFrame {
  /*
  According to the Swing 1.0.1
  documentation, the following
  statement will:
  "Reference an array of objects that
  provide some information about the
  LookAndFeel implementations that have
  been installed with this java
  development kit.  The LookAndFeelInfo
  objects can be used by an application
  to construct a menu of look and feel
  options for the user or to set the
  look and feel at start up time."
  */
  UIManager.LookAndFeelInfo[]
                        plafInfoArray;
  //---------------------------------//
  //save a reference to the test GUI
  // here
  JFrame testGui;
  //ref to this object
  PlafPanel03 thisPlafPanel = this;

  //constructor
  public PlafPanel03(JFrame testGui) {
    //save ref to test GUI
    this.testGui = testGui;
    this.getContentPane().add(
            new JLabel("=========="
             + "Plaf Selection Panel, "
             + "R.G.Baldwin"
             + "=========="));

    //Get the list of L&Fs installed
    // in the current system.
    plafInfoArray =
        UIManager.
            getInstalledLookAndFeels();

    //Create a vector of references to
    // JButton objects with one element
    // in the vector for each L&F
    // implementation in the system
    Vector theButtons =
      new Vector(plafInfoArray.length);

    //Create one JButton object for
    // each L&F implementationand put
    // its reference in the Vector.
    for(int cnt = 0;
            cnt < plafInfoArray.length;
                cnt++){
      theButtons.addElement(
                        new JButton());

      //Get the name of the class for
      // each specific L&F
      // implementation
      String theClassName =
                    plafInfoArray[cnt].
                        getClassName();

      //Extract a short name for each
      // specific L&F implementation
      // and use it to label the
      // JButton corresponding to that
      // L&F. The short name appears
      // following the last period in
      // the String representation of
      // the class name for the L&F
      // implementation. Note the
      // requirement to downcast the
      // reference extracted from the
      // Vector.
      String label =
            theClassName.substring(
              theClassName.lastIndexOf(
                                ".")+1);
      ((JButton)theButtons.
         elementAt(cnt)).setText(label);

      //Add an action listener to each
      // JButton that will cause the
      // L&F to change to the one
      // represented by that JButton
      // whenever the JButton is
      // clicked. Note that because the
      // references to the buttons are
      // stored in a Vector, it is
      // necessary to downcast them
      // from Object to JButton.
      ((JButton)theButtons.
           elementAt(cnt)).
             addActionListener(
               new MyActionListener(
                       theClassName));

      //Add each JButton to the JFrame
      this.getContentPane().
               add((JButton)theButtons.
                       elementAt(cnt));
    }//end for loop

    this.getContentPane().
           setLayout(new FlowLayout());
    this.setTitle(
          "Plaf Panel by R.G.Baldwin");
    this.setBounds(30,300,350,150);
    this.setVisible(true);
  }//end constructor
  //=================================//

  //Inner class for action listeners
```

*Continued on next page.*

```
  this.getContentPane().
    add((JButton)theButtons.
elementAt(cnt));
}//end for loop
```

There is a little more work required to polish the JFrame object. This is standard stuff and doesn't really relate to PLAF, but I'm including it for continuity. This codes ends the constructor:

```
  this.getContentPane().
          setLayout(new FlowLayout());
  this.setTitle(
         "Plaf Panel by R.G.Baldwin");
  this.setBounds(30,300,350,150);
  this.setVisible(true);
}//end constructor
```

This code shows the beginning of an inner class used to instantiate the ActionListener objects. The constructor for the listener object saves the name of the L&F class associated with the button on which the listener is registered:

**Components that are properly spaced in one L&F implementation might appear overcrowded in another**

```
class MyActionListener
              implements ActionListener{
  String plafClassName;

  MyActionListener(//constructor
    String plafClassName){
    //save the incoming parameter
    this.plafClassName = plafClassName;
  }//end constructor
```

The `actionPerformed()` method is the heart of all ActionListener objects. This code shows the beginning of this method, which invokes the static `setLookAndFeel()` method of the UIManager class to set the current default L&F. Note that this method is not being invoked on any specific object. Rather, it is being invoked in a general sense, which sets the default L&F for the entire application. However, just setting the L&F doesn't make it visually effective. Some additional work is required later on to accomplish that. This method throws several different exceptions so it is enclosed in a try/catch block:

```
public void actionPerformed(
  ActionEvent e){
  //Set the current default L&F to
  // that passed in as a parameter
  // to the constructor
  try{
    UIManager.setLookAndFeel(
      plafClassName);
  }catch(Exception ex){
    System.out.println(ex);}
```

Simply setting the current default L&F doesn't cause the visual image to change. It is necessary to take specific action to cause the display to reflect the new L&F. The code accomplishes this by invoking the static `updateComponentTreeUI()` method of the SwingUtilities class on two different references: the test GUI and the PLAF Panel. It will cause both the test GUI and the PLAF Panel to reflect the new L&F selected by clicking on one of the buttons:

```
    SwingUtilities.
      updateComponentTreeUI(
      thisPlafPanel);
    SwingUtilities.
      updateComponentTreeUI(testGui);
  }//end actionPerformed()
}//end class MyActionListener
```

That is the end of the definition of the `actionPerformed()` method, the end of the ActionListener class. The only code remaining is a short `main()` method used to test the class on a stand-alone basis. (It is frequently useful to test a class to confirm that it works properly before hooking it to other classes.) Recall, however, that objects of this class are not intended for use in a stand-alone mode. They should be hooked to a GUI under test and used to modify the L&F of that GUI by selecting each of the L&F implementations currently installed in the system. This makes it possible for you to easily view your GUI for each L&F implementation:

```
  public static void main(String[] args){
    new PlafPanel03(new JFrame());
  }//end main()
}//end class PlafPanel03
```

Note that if you run this class in a stand-alone mode, you will need to press Ctrl+C to terminate it because there is no listener object registered on the close button on the JFrame. ■

## Code Online

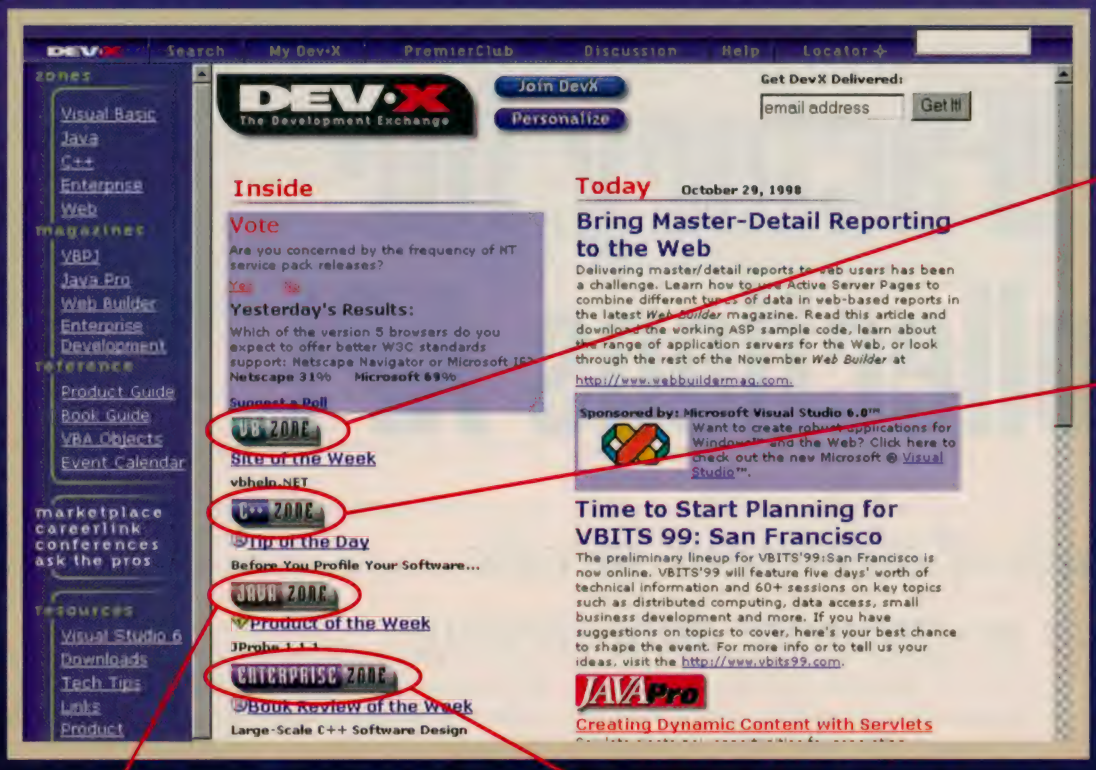
All the code in this article is available for free at www.java-pro.com. Use the Locator+ codes shown below to download. Free online registration is required. See the Online Resources box in "Letters to the Editor" for more information.

***A Test Panel for Pluggable Look and Feel***
Registered Locator+: **JP0299**
Premier Locator+: **JP0299R1**

*Continued from previous page.*

```
class MyActionListener
          implements ActionListener{
  //save name of plaf class here
  String plafClassName;

  //Constructor
  MyActionListener(
                String plafClassName){
    //save the incoming parameter
    this.plafClassName =
                          plafClassName;
  }//end constructor

  public void actionPerformed(
                          ActionEvent e){
    //Set the current default L&F to
    // that passed in asa parameter
    // to the constructor
    try{
      UIManager.setLookAndFeel(
                          plafClassName);
    }catch(Exception ex){
            System.out.println(ex);}

    //Now cause the current default
    // L&F to take effect. The
    // description of the
    // updateComponentTreeUI() method
    // as extracted from the Swing
    // 1.0.1 documentation is as
    // follows:

    //"A simple minded look and feel
    // change: ask each node in the
    // tree to updateUI(), i.e. to
    // initialize its UI property
    // with the current look
    // and feel."

    //Set the L&F for this PlafPanel
    // object
    SwingUtilities.
                updateComponentTreeUI(
                      thisPlafPanel);
    //Set the L&F for the test GUI
    // object
    SwingUtilities.
                updateComponentTreeUI(
                             testGui);

  }//end actionPerformed()
}//end class MyActionListener

//main method for stand-alone testing
public static void main(
                          String[] args){
  new PlafPanel03(new JFrame());
}//end main()
}//end class PlafPanel03
//=====================================//
```

# www.devx.com


Search
My DevX
PremierClub
Discussion
Help
Locator
zones
Visual Basic
Java
C++
Enterprise
Web
magazines
VBPJ
Java Pro
Web Builder
Enterprise Development
reference
Product Guide
Book Guide
VBA Objects
Event Calendar
marketplace
careerlink
conferences
ask the pros
resources
Visual Studio 6
Downloads
Tech Tips
Links
Product
DEVX
The Development Exchange
Join DevX
Personalize
Get DevX Delivered:
email address
Get It!
Inside
Vote
Are you concerned by the frequency of NT service pack releases?
Yesterday's Results:
Which of the version 5 browsers do you expect to offer better W3C standards support: Netscape Navigator or Microsoft IE?
Netscape 31%
Microsoft 69%
Suggest a Poll
VB ZONE
Site of the Week
vbhelp.NET
C++ ZONE
Tip of the Day
Before You Profile Your Software...
JAVA ZONE
Product of the Week
JProbe 3.1.1
ENTERPRISE ZONE
Book Review of the Week
Large-Scale C++ Software Design
Today
October 29, 1998
Bring Master-Detail Reporting to the Web
Delivering master/detail reports to web users has been a challenge. Learn how to use Active Server Pages to combine different types of data in web-based reports in the latest Web Builder magazine. Read this article and download the working ASP sample code, learn about the range of application servers for the Web, or look through the rest of the November Web Builder at
http://www.webbuildermag.com.
Sponsored by: Microsoft Visual Studio 6.0™
Want to create robust applications for Windows® and the Web? Click here to check out the new Microsoft ® Visual Studio™.
Time to Start Planning for VBITS 99: San Francisco
The preliminary lineup for VBITS'99:San Francisco is now online. VBITS'99 will feature five days' worth of technical information and 60+ sessions on key topics such as distributed computing, data access, small business development and more. If you have suggestions on topics to cover, here's your best chance to shape the event. For more info or to tell us your ideas, visit the http://www.vbits99.com.
JAVA Pro
Creating Dynamic Content with Servlets

**The VB Zone:**
www.vb-zone.com

**The C++ Zone:**
www.cplus-zone.com

Each DevX Zone is a Web site, targeting your specific information needs for techniques and tools, from VB to Java, from C++ to SQL.

**The Java Zone:**
www.java-zone.com

**The Enterprise Zone:**
www.enterprise-zone.com

# Need programming information, NOW?

## It's all here on The Development Exchange (DevX)

**DevX's family of Web sites is your most comprehensive source for programming information available.** Get immediate access to essential tools, code, and resources aimed at helping you solve your toughest programming challenges. And now, whether you're undertaking intranet solutions with Java, end-user applications with VB, or ISV products with C++, **you're just a click away from the right tool in one of DevX's four development zones.**

- **Technical articles and code** from *Visual Basic Programmer's Journal*, *Enterprise Development*, and *Java™Pro* magazines
- **Daily programming tips**—plus a library of thousands more
- **Ask the Pros**—get answers to tough development questions
- **Product Reviews**—hundreds of reviews updated weekly
- **Forums**—dedicated to development
- **40% off Books** of the Week—plus editors' picks and reviews of top programming titles
- **Search Products**—from our database of over 5,000 programming tools
- **Conference updates**—register online

**And that's just the beginning.**

DEVX The Development Exchange

Explore the rest yourself at
**www.devx.com**

Microsoft, Windows, Visual Basic, Visual C++ are registered trademarks of Microsoft Corporation. Java is a registered trademark of Sun Microsystems.
The Development Exchange, DevX, CareerLink, and Ask the Pros are trademarks of Fawcette Technical Publications.
Fawcette Technical Publications, 209 Hamilton Avenue, Palo Alto, CA 94301-2500 Phone: 650.833.7100 Fax: 650.853.0230 www.devx.com

# A SortFactory for the Collections API

## *Take advantage of algorithm-specific characteristics to optimize data sorting*

by Claude Duguay

Although the Java 2 Collections API provides a number of useful data structures and algorithms, it provides only one static method for sorting data in the Collections class. This article will show you how to extend the basic API with a SortFactory, providing a sorting algorithm for use with any of the sortable collections. The general-purpose heap sort provided by the Java 2 API is a good choice in most cases, but this group of algorithms offers more flexibility for optimization.

This article will also illustrate a number of useful techniques for extending Java. We'll apply a few programming patterns, including the Factory and Singleton patterns, basing our approach on a Sort interface, which makes the sorting algorithms interchangeable (see Figure 1). Finally, we'll capitalize on the Comparable and Comparator interfaces provided in the Java Collections API to develop a fully generalized implementation that can be used with any List or Object array.

The Java Collections API provides a set of interfaces and concrete implementations that allow you to manipulate object groups. The general objective of the interfaces is to present a unified architecture for representing and manipulating collections, independent of the details of their representation. The interfaces support interoperability between unrelated APIs, and reduce design and learning effort, encouraging reuse:

- **Collection** is a group of objects in no particular order.
- **Set** is a collection with no duplication, in no particular order.
- **List** is an ordered collection with possible duplicates.
- **Map** is collection of key/value pairs with a maximum of one value per key.
- **SortedSet** is a Set that is always kept sorted.
- **SortedMap** is a Map that is always kept sorted.

The API provides six concrete implementations that offer high-performance solutions:

- **HashSet** implements the Set interface using a hash table.
- **TreeSet** implements the Set interface using a red-black tree structure.

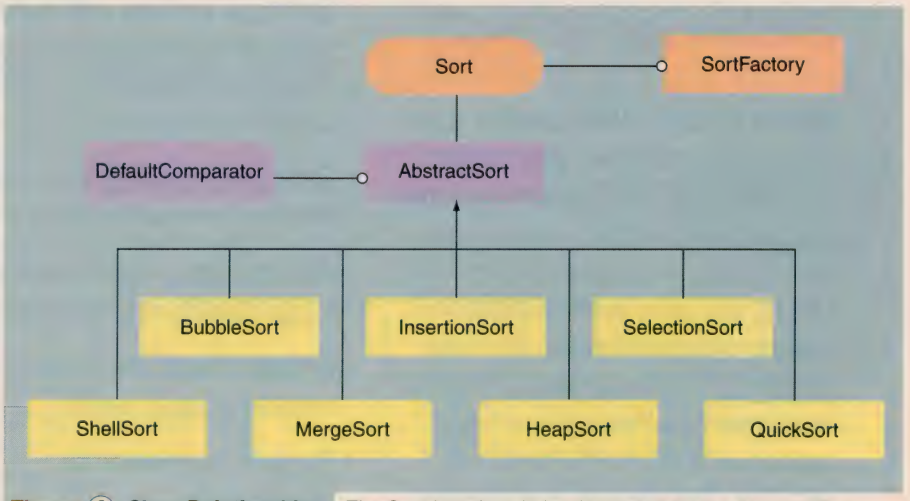



**Figure 1 Class Relationships** The Sort interface is implemented by the AbstractSort class, which is the parent of each of the other sorting classes. AbstractSort provides a DefaultComparator. The SortFactory generates requested Sort classes on demand.

Claude Duguay manages a software-development team at Atrieva Corp. Claude has more than 17 years of experience in programming and virtually all aspects of building a software-development company. Reach him at claude@atrieva.com.

| Algorithm | Strategy | Random | Ascending | Descending | Equality |
|---|---|---|---|---|---|
| Bubble | Transposition | O(N^2) | O(N) | O(N^2) | O(N) |
| Selection | Priority Queue | O(N^2) | O(N^2) | O(N^2) | O(N^2) |
| Insertion | Insert and Keep Sorted | O(N^2) | O(N) | O(N^2) | O(N) |
| Shell | Diminishing Increment | O(N^1.25) | O(N log N) | O(N^1.5) | O(N log N) |
| Quick | Divide and Conquer | O(N log N) | O(N log N) | O(N log N) | O(N log N) |
| Merge | Divide and Conquer | O(N log N) | O(N log N) | O(N log N) | O(N log N) |
| Heap | Divide and Conquer | O(N log N) | O(N log N) | O(N log N) | O(N) |

**Table 1 Benchmark Results** We averaged sort times for different sample sets. The samples are randomized, already in ascending or descending order, or the elements are equal to each other.

- **ArrayList** implements the List interface with an unsynchronized Vector.
- **LinkedList** implements the List interface with a doubly linked list (a chain).
- **HashMap** implements the Map interface with a hash table.
- **TreeMap** implements the SortedMap interface with a red-black tree.

In Java 2, the Vector class now implements the List interface and the Hashtable class implements the Map interface. These two classes remain synchronized, but the new collection classes are not, making them faster than their predecessors in Java 1.1. It's possible to ensure thread safety by calling the static `synchronizedCollection()` method in the Collections class. You can also make a collection immutable by calling the static `unmodifiableCollection()` method.

To make arrays easier to manage, the Arrays class implements an `asList()` method that lets you manipulate the array as if it were a List. Several algorithms are provided to sort, shuffle, search, reverse, fill, copy, and find minimum or maximum values. You'll find these implemented as static methods in the Collections class. Both the Collections and Array classes have a large number of useful methods you'll want to be familiar with.

The designers of the Collections API have provided a handful of abstract classes, which implement several useful methods, allowing you to leverage this foundation when you are forced to build your own collection classes. This is rarely necessary, but there are cases when you might want to optimize for specific conditions (such as building a disk-based hash table or an alternate tree implementation). The abstract classes provide skeletal implementations for the various interfaces:

- **AbstractCollection** implements the Collection interface.
- **AbstractSet** implements the Set interface.
- **AbstractList** implements the List interface.
- **AbstractSequentialList** implements an ordered list.
- **AbstractMap** implements the Map interface.

The Collections API was designed to be relatively thin, compact, and as uncomplicated as possible. Many of the existing Java classes that made use of the Vector and/or Hashtable can use the new Collections classes, and I highly recommended this approach be applied to your development as early as possible. The new collections are faster, more flexible, and represent the better choice in most cases.

One of the big surprises in this API is the UnsupportedOperationException, which can be thrown by unimplemented methods. When attempting to capture all the subtleties and possible variations in different data structures, it's easy to find yourself implementing a wide array of complicated interfaces. Rather than produce a large number of interfaces, the designers of the Java Collections API decided to consider the non-core methods optional, throwing a suitable exception when they are unavailable. Although this conflicts with the whole idea of declaring interfaces to define roles and imperative methods, it was considered a reasonable compromise in order to keep the API as simple as possible without unnecessarily constraining it.

**Creating classes only when necessary is often referred to as "lazy instantiation"**

## Defining a Sort Interface

The Comparable interface defines the `compareTo()` method that lets you determine relative order for the object being compared. The Comparator interface defines a `compare()` method, which does the same thing as `compareTo()` but acts on two independent objects, along with the `equals()` method for determining equality between Comparators.

To make our sorting implementations uniform, we define a Sort interface, which forces these methods to be present:

```
import java.util.List;
import java.util.Comparator;

public interface Sort
{
  public void sort(List list);
  public void sort(Comparable[] list);
  public void sort(
    Object[] list, Comparator comp);
}
```

By having the first two methods call the more-flexible third method, we can reduce the amount of code in each of the sorting classes. We implement an AbstractSort class that each of the concrete classes can extend. In addition, we define a DefaultComparator, which is always used in cases where you don't explicitly define another class by using the extended method syntax.

To select algorithms, we implement a factory class called SortFactory. The Factory pattern is designed to produce instances of the required classes on demand. We'll use a static `getInstance()` method with a named argument that returns an implementation of the Sort interface. To maximize efficiency, only one instance is actually created for each of the algorithms, and these are only created when asked for the first time.

### Sorting Algorithms

Over the years, numerous sorting algorithms have been developed and studied by researchers and computing professionals. To gauge their efficiency, the O notation was developed to measure the growth rate associated with algorithms applied to data sets of different sizes. The average number of operations required to sort a list of elements plays a direct role in this equation. In these equations, N represents the number of elements in the list being sorted. The fastest algorithms run in linear O(N) time, the weaker ones typically use an inner and outer loop and run in O(N^2) time.

**The Collections and Array classes have a large number of useful methods you'll want to be familiar with**

Table 1 shows how different data conditions can affect these algorithms. From these numbers, you can conclude that the Bubble and Insertion algorithms do well if the data is neatly sorted or uniform, but they perform poorly in other cases. The Heap sort performs well overall and excels on sets that are fairly uniform. Keep in mind that these assessments are based on extreme conditions, which rarely occur in the real world. Determining which of these algorithms will meet your needs might require benchmarking measurements and an in-depth understanding of your particular data conditions.

Here's a quick overview of the algorithms and how they function:

- **BubbleSort** makes a series of passes through the collection, exchanging adjacent items if they are in the incorrect order. The sort procedure ends when a full pass is made and no exchanges were necessary.
- **SelectionSort** searches the collection for the smallest item and swaps it with the currently marked lower bound. The lower bound is moved successively up the collection until the final element is in place.
- **InsertionSort** picks the item right after the current lower bound and finds the correct position for it in the preceding part of the list, moving entries above the insertion up by one position.
- **ShellSort** is similar to InsertionSort, but it tries to sort elements a fixed distance apart, reducing the distance over several iterations. This minimizes the number of elements that need to be moved when an item is inserted.
- **QuickSort** divides the collection into two halves around a pivotal element and moves the elements above the pivot to the first half and elements that are larger in the second half. The process is applied recursively to each half until the elements are in order.
- **MergeSort** divides a list into sorted subsections by first taking every sequential pair of elements, sorting them, and then grouping every pair of pairs and merging them in the correct order, repeating the process until the series is in order.
- **HeapSort** uses the heap data structure, which represents a tree by dividing a list into successive binary divisions. The algorithm inserts each element into the tree and removes them successively, reorganizing the tree as elements are positioned at the end of the list.

### Supporting Classes

The AbstractSort class in Listing 1 provides a foundation for building the concrete sort classes. We use the DefaultComparator to do comparisons if none are explicitly provided by calling the third sort method variant (see Listing 2). The DefaultComparator implements the Comparator interface. We test the two arguments in the `compare()` method and throw an

## The Big O

Algorithm analysis provides a framework for measuring the efficiency of algorithms. Although it is impossible to perform a deep scientific analysis that takes into account all possible machine, memory, language, and implementation-specific issues, a rough estimate, based on the most significant elements affecting performance, is important when deciding which algorithm to use. The O notation, a.k.a. Big-O or Asymptotic notation, effectively eliminates the less significant elements of the equation. The objective is to produce a simplified formula, which is similar to, less than, or equal to the actual algorithm performance curve.

A formula that reads: f(n) = XN^2 + YN + Z; factors into something like: O(N^2). Intuitively it's easy to understand that N^2 (N squared) is the most significant factor in the equation. Removing any other element has less impact on the curve. There are other variants to the Big O—Big Omega, Big Theta, and so on—that measure growth rate by applying slightly different formulas. Based on this explanation, you can probably guess where Big O got it's name. The O stands for Order. So the Bubble sort, which includes a pair of nested loops, is said to perform on the order of N squared.

To make things easier to compare, algorithms are usually grouped into complexity classes. This table shows the most common classes. Notice that the general purpose sorting algorithms fall between Quadratic and Linear. For obvious reasons, it is not wise to use the slower algorithms on large sets of data. It's also good idea to understand enough about these factors to predict performance in your applications, whether these factors apply to sorting or other algorithms you might develop or use:

| Name | Notation |
|---|---|
| Constant | O(1) |
| Logarithmic | O(log N) |
| Linear | O(N) |
| N log N | O(N log N) |
| Quadratic | O(N^2) |
| Cubic | O(N^3) |
| Exponential | O(2^N) |

Keep in mind that O notation is an indicator that cannot and does not account for memory usage, resource constraints, machine speed, or other elements, so your results will vary widely. Obviously, a faster machine will run the algorithm more quickly, but certain algorithms perform better in constrained memory environments (like Heap sort, which consumes less memory), while others might be better in slower memory conditions (Merge sort can be applied to disk-based sorting). Finally, the order of the data can affect performance, especially in sorting. Some algorithms move less data, and the O notation cannot account for those conditions.

—*C.D.*

IllegalArgumentException if they do not implement the Comparable interface. If they do, we cast the arguments and call the Comparable interface `compareTo()` method. The `equals()` method simply calls `compare()` and returns true if the returned value is zero (0).

The SortFactory code in Listing 3 is the heart of our implementation. The `getInstance()` method returns an instance of a sorting class that implements the Sort interface. The SortFactory is implemented as a Singleton. The constructor is declared as private, so no other instance can be created. We use a static Singleton variable to hold the only permissible instance of this class. The `getInstance()` method is declared as static and uses the Singleton instance to create suitable Sort classes.

The `getInstance()` method always checks to see if an instance of the requested Sort class has already been created, returning it if it finds one. Otherwise, the class is created, and a reference is saved in a suitably named static variable. This technique of creating classes only

**Listing ① AbstractSort** The AbstractSort class provides a foundation for building the concrete sort classes.

```
import java.util.List;
import java.util.Comparator;

public abstract class AbstractSort implements Sort
{
  private static DefaultComparator
    defaultComparator = new DefaultComparator();

  public void sort(List list)
  {
    int size = list.size();
    Comparable[] array = new Comparable[size];
    for (int i = 0; i < size; i++)
    {
      array[i] = (Comparable)list.get(i);
    }
    sort(array);
    for (int i = 0; i < size; i++)
    {
      list.set(i, array[i]);
    }
  }

  public void sort(Comparable[] array)
  {
    sort(array, defaultComparator);
  }

  public abstract void sort(Object[] list, Comparator comp);
}
```

**Listing ② DefaultComparator** We test the two arguments in the compare() method and throw an IllegalArgumentException if they do not implement the Comparable interface.

```
import java.util.Comparator;

public class DefaultComparator implements Comparator
{
  public int compare(Object one, Object two)
  {
    if (!(one instanceof Comparable
      && two instanceof Comparable))
      throw new IllegalArgumentException(
        "DefaultComparator requires Comparable elements");
    Comparable left = (Comparable)one;
    Comparable right = (Comparable)two;
    return left.compareTo(right);
  }

  public boolean equals(Object one, Object two)
  {
    return compare(one, two) == 0;
  }
}
```

**Listing ③ SortFactory** The SortFactory is implemented as a Singleton.

```
import java.util.Comparator;

public class SortFactory
{
  private static SortFactory singleton = new SortFactory();
  private static Sort bubble, insertion,
    selection, shell, merge, heap, quick;

  private SortFactory() {}

  public static Sort getInstance(String name)
  {
    if (name.equalsIgnoreCase("Bubble"))
    {
      if (singleton.bubble == null)
        singleton.bubble = new BubbleSort();
      return singleton.bubble;
    }
    if (name.equalsIgnoreCase("Insertion"))
    {
      if (singleton.insertion == null)
        singleton.insertion = new InsertionSort();
      return singleton.insertion;
    }
    if (name.equalsIgnoreCase("Selection"))
    {
      if (singleton.selection == null)
        singleton.selection = new SelectionSort();
      return singleton.selection;
    }
    if (name.equalsIgnoreCase("Merge"))
    {
      if (singleton.merge == null)
        singleton.merge =new MergeSort();
      return singleton.merge;
    }
    if (name.equalsIgnoreCase("Shell"))
    {
      if (singleton.shell == null)
        singleton.shell =new ShellSort();
      return singleton.shell;
    }
    if (name.equalsIgnoreCase("Heap"))
    {
      if (singleton.heap == null)
        singleton.heap =new HeapSort();
      return singleton.heap;
    }
    if (name.equalsIgnoreCase("Quick"))
    {
      if (singleton.quick == null)
        singleton.quick = new QuickSort();
      return singleton.quick;
    }
    throw new IllegalArgumentException(
      "Sorting algorithm " + name + " is not supported");
  }
}
```

**Listing ④ SortingTest** The SortingTest class implements the harness. The randomList() method generates a List collection with the specified number of randomized elements.

```
import java.util.List;
import java.util.ArrayList;

public class SortingTest
{
  public static List randomList(int length)
  {
    List list = new ArrayList();
    for (int i = 0; i < length; i++)
    {
      list.add(new Integer((int)(Math.random() * 99)));
    }
    return list;
  }

  public static void printList(List list)
  {
    for (int i = 0; i < list.size(); i++)
    {
      Object element = list.get(i);
      System.out.print(element + " ");
    }
    System.out.println();
  }

  public static List copyList(List list)
  {
    List copy = new ArrayList();
    for (int i = 0; i < list.size(); i++)
    {
      int value = ((Integer)list.get(i)).intValue();
      copy.add(new Integer(value));
    }
    return copy;
  }

  public static boolean sortedList(List list)
  {
    for (int i = 0; i < list.size() - 1; i++)
    {
      Comparable item = (Comparable)list.get(i);
      Comparable next = (Comparable)list.get(i + 1);
      if (item.compareTo(next) > 0) return false;
    }
    return true;
  }

  public static void main(String[] args)
  {
    String[] algorithms = {"Bubble", "Insertion",
      "Selection", "Shell", "Merge", "Quick", "Heap"};
    List list = randomList(20);
    boolean success = true;
    int count = 20;
    for (int j = 0; j < count; j++)
    {
      for (int i = 0; i < algorithms.length; i++)
      {
        List copy = copyList(list);
        Sort sorter = SortFactory.getInstance(algorithms[i]);
        String className = sorter.getClass().getName();
        if (!className.startsWith(algorithms[i]))
          throw new IllegalArgumentException(
            "Algorithm failed");
        sorter.sort(copy);

        System.out.println();
        System.out.println("TESTING  : " + className);
        System.out.print("Original : ");
        printList(list);
        System.out.print("Sorted   : ");
        printList(copy);
        System.out.print("RESULT   : ");
        boolean result = sortedList(copy);
        System.out.println(result ? "SUCCESS" : "FAILURE");
        if (!result) success = false;
      }
      System.out.print("\nLOOP " + (j + 1) + " RESULT: ");
      System.out.println(success ? "SUCCESS" : "FAILURE");
    }
    System.out.print(
      "\nTOTAL " + count + " LOOP TEST RESULT: ");
    System.out.println(success ? "SUCCESS" : "FAILURE");
  }
}
```

**Listing ⑤ SortingBenchmark** The benchmark performs a basic test based on random lists of a given number of elements.

```
import java.io.PrintStream;
import java.util.ArrayList;
import java.util.List;

public class SortingBenchmark
{
  public static List randomList(int length)
  {
    List list = new ArrayList();
    for (int i = 0; i < length; i++)
    {
      list.add(new Integer((int)(Math.random() * 99)));
    }
    return list;
  }

  public static List copyList(List list)
  {
    List copy = new ArrayList();
    for (int i = 0; i < list.size(); i++)
    {
      int value = ((Integer)list.get(i)).intValue();
      copy.add(new Integer(value));
    }
    return copy;
  }

  public static long mark()
  {
    return System.currentTimeMillis();
  }

  public static long time(long mark)
  {
    return System.currentTimeMillis() - mark;
  }

  public static void main(String[] String_1darray1)
  {
    String[] algorithm = {"Bubble", "Insertion",
      "Selection", "Shell", "Merge", "Quick", "Heap"};
    int iterations = 5;
    int[] elements = { 100, 500, 1000, 5000, 10000, 50000 };
    long[][] averages =
      new long[elements.length][algorithm.length];

    for(int i = 0; i < iterations; i++)
    {
      System.out.print( "\nIteration: " + (i + 1));
      for(int j = 0; j < elements.length; j++)
      {
        System.out.print( "\n" + elements[j] + " elements" );
        List list = randomList(elements[j]);
        int count = 0;
```

*Listing continued on next page.*

# RECEIVE YOUR FREE SUBSCRIPTION

to ***ENT*ERPRISE DEVELOPMENT**—the essential tool for the management and architecture of application development projects.

I wish to receive ENTERPRISE DEVELOPMENT Magazine free of charge! ☐ Yes! ❑ No

| | | | |
|---|---|---|---|
| NAME | | TITLE | |
| SIGNATURE (REQUIRED) | X | COMPANY | |
| DATE (REQUIRED) | / / | DEPARTMENT | |
| PHONE | ( ) | ADDRESS | |
| FAX | ( ) | CITY, STATE | |
| E-MAIL ADDRESS | | ZIP/POSTAL CODE / COUNTRY | |

**Annual subscription rate is $99.95 for nonqualified subscribers.**

PLEASE ANSWER ALL QUESTIONS ON BOTH SIDES, SIGN AND DATE THE CARD. INCOMPLETE FORMS CANNOT BE PROCESSED.

**1. PRIMARY BUSINESS ACTIVITY AT YOUR LOCATION (SELECT ONLY ONE):**

01 ❑ Finance, banking, accounting, insurance
02 ❑ Manufacturing other than computer hardware and software
03 ❑ Manufacturing computers and computer software
04 ❑ Non-computer retailer, distributor, wholesaler
05 ❑ Applications software development
06 ❑ Aerospace
07 ❑ Government, military
08 ❑ Agriculture, mining, construction, oil & gas
09 ❑ Transportation, utilities
10 ❑ Media, marketing, advertising
11 ❑ Real estate
12 ❑ Research and development
13 ❑ Engineering
14 ❑ VAR/Systems Integrators
15 ❑ Communications services
16 ❑ Computer-related retailer, distributor, wholesaler
17 ❑ Systems software development
18 ❑ Education
19 ❑ Health, medical, legal
20 ❑ Consulting (please specify)_____
99 ❑ Other (please specify) _____

**2. WHAT IS YOUR CURRENT JOB FUNCTION? (SELECT ONLY ONE FROM THE FOLLOWING SIX (6) CATEGORIES):**

Application development manager/architect:
01 ❑ Applications development manager
02 ❑ Software development director
03 ❑ Software architect
04 ❑ Software project leader
05 ❑ Project manager
06 ❑ Advanced technology evaluation
07 ❑ Software team leader
08 ❑ Software engineering supervisor
09 ❑ Systems analyst/designer
10 ❑ Database designer
11 ❑ Application management/ administration
99 ❑ Other (please specify) _____

IS management:
30 ❑ CIO, Technical VP
31 ❑ IS/MIS manager
99 ❑ Other (please specify) _____

Application development staff:
40 ❑ Applications developer/programmer
99 ❑ Other (please specify) _____

Computer operations staff:
50 ❑ System Administrator
51 ❑ LAN Administrator
52 ❑ Help Desk
99 ❑ Other (please specify) _____

Corporate management:
60 ❑ CEO, president, partner, chairman, owner
61 ❑ VP, controller, director of marketing/sales, General manager, division/department head

Other (categories not listed above):
99 ❑ Other (please specify)

_____

**3. ARE YOU DEVELOPING APPLICATIONS THAT INVOLVE WINDOWS NT OR DO YOU PLAN TO BE IN THE NEXT 12 MONTHS?**

01 ❑ Yes 02 ❑ No

**4. ARE YOU DEVELOPING APPLICATIONS THAT WILL INVOLVE ONE OR MORE SERVERS OR HOSTS OR DO YOU PLAN TO BE IN THE NEXT 12 MONTHS?**

01 ❑ Yes 02 ❑ No

**5. DO YOU EVALUATE, SPECIFY, RECOMMEND, AUTHORIZE, OR APPROVE THE PURCHASE OF (PLEASE CHECK ALL THAT APPLY):**

01 ❑ Application development tools?
02 ❑ Database products?
03 ❑ Internet and/or Intranet products?
04 ❑ Middleware products?
05 ❑ Hardware servers?
06 ❑ Modeling tools
07 ❑ Testing/QA tools
08 ❑ Software process tools (Project/Process Management, Metrics, CM, etc.)
99 ❑ None of the above

**6. WHICH OF THE FOLLOWING SOFTWARE SERVERS DO YOU BUY, SPECIFY, RECOMMEND, OR APPROVE THE PURCHASE OF AND/OR CURRENTLY HAVE INSTALLED AT THIS LOCATION (PLEASE CHECK ALL THAT APPLY):**

| | Buy, Specify, Recommend, Approve | Installed |
|---|---|---|
| Web | 01 ❑ | 02 ❑ |
| Database | 03 ❑ | 04 ❑ |
| Application | 05 ❑ | 06 ❑ |
| Transaction | 07 ❑ | 08 ❑ |
| E-mail | 09 ❑ | 10 ❑ |
| Imaging | 11 ❑ | 12 ❑ |
| Commerce | 13 ❑ | 14 ❑ |
| Remote Access | 15 ❑ | 16 ❑ |
| Communication/SNA | 17 ❑ | 18 ❑ |

99 ❑ None of the above

(over)

For faster service fax both sides of this form to: **1-650-321-3818**
OR fill out the electronic form at www.enterprisedev.com

**7. SERVER/HOST OPERATING SYSTEMS IN USE AT THIS ENTIRE LOCATION (PLEASE CHECK ALL THAT APPLY):**

01 ❑ Windows NT Server
02 ❑ Unix (all flavors)
03 ❑ Netware/Intranetware
04 ❑ OS/400
05 ❑ VMS
06 ❑ MVS/VM
07 ❑ OS/2
99 ❑ Other (please specify) _____

**8. WHAT IS THE ANNUAL DOLLAR VALUE OF COMPUTER PRODUCTS AND SERVICES FOR WHICH YOU ARE PERSONALLY INVOLVED IN THE PURCHASING DECISIONS FOR YOUR ORGANIZATION'S OR LARGEST CLIENT'S INTERNAL USE?**

01 ❑ More than $10 million
02 ❑ $5 million - $9,999,999
03 ❑ $2 million - $4,999,999
04 ❑ $1 million - $1,999,999
05 ❑ $500,000 - $999,999
06 ❑ $100,000 - $499,999
07 ❑ $50,000 - $99,999
08 ❑ $10,000 - $49,999
09 ❑ Less than $10,000

**9. WHICH OF THE FOLLOWING CATEGORIES OF PRODUCTS DO YOU RECOMMEND, EVALUATE, SPECIFY, BUY, OR APPROVE THE PURCHASE OF? (CHECK ALL THAT APPLY):**

01 ❑ Back office suites
02 ❑ N-Tier Capable Development Tools
03 ❑ ORBs
04 ❑ RPC middleware
05 ❑ Message Oriented Middleware
06 ❑ Application Integration Software
07 ❑ TP Monitors or Transaction Servers
08 ❑ Application Servers
09 ❑ Application Management Software
10 ❑ Component software (VBX, OCX, ActiveX, custom controls, JavaBeans, etc.)
11 ❑ Configuration Management/Version Control
12 ❑ Project Management Tools
13 ❑ Process Management Tools
14 ❑ Estimation/planning/metrics tools
15 ❑ Class Libraries
16 ❑ Application Frameworks
17 ❑ C/C++ Tools
18 ❑ Java Tools
19 ❑ VB Tools
20 ❑ 4GL Tools
21 ❑ OLAP, Data Mining or Data Warehousing tools
22 ❑ Reporting Tools
23 ❑ Web Authoring Tools
24 ❑ Expert systems/rules engines
25 ❑ Repository
26 ❑ Security Tools/Firewall Software
99 ❑ Other (please specify) _____
99 ❑ None of the above

**10. DO YOU RECOMMEND, SPECIFY, APPROVE OR PURCHASE APPLICATION PACKAGES SUCH AS THOSE SOLD BY SAP, PEOPLESOFT, AND BAAN?**

01 ❑ Yes 02 ❑ No

If YES, which types? (Check ALL that apply):

01 ❑ ERP Suites
02 ❑ Human Resources
03 ❑ Sales Automation
04 ❑ Financial/Accounting
05 ❑ Retail/Point of Purchase
06 ❑ Manufacturing/Inventory
07 ❑ Supply Chain Management
99 ❑ Other (please specify) _____

**11. DO YOU RECOMMEND, SPECIFY, APPROVE OR PURCHASE APPLICATION DEVELOPMENT TOOLS FOR CUSTOMIZATION OF AND INTEGRATION WITH ERP OR OTHER APPLICATION PACKAGES?**

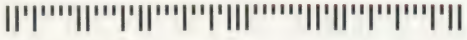
01 ❑ Yes 02 ❑ No

**12. WHAT IS THE NUMBER OF EMPLOYEES OF THE LARGEST BUSINESS ENTITY BY WHOM YOU ARE EMPLOYED?**

01 ❑ Under 100
02 ❑ 100 - 999
03 ❑ 1,000 - 4,999
04 ❑ 5,000 - 9,999
05 ❑ 10,000 - 49,999
06 ❑ More than 49,999

*Publisher reserves the right to determine qualification for FREE subscriptions.*

▼ FOLD HERE ▼ ▼ FOLD HERE ▼



NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO 317 PALO ALTO CA

POSTAGE WILL BE PAID BY ADDRESSEE

ENTERPRISE DEVELOPMENT

Attn: New Subscriptions
209 HAMILTON AVENUE
PALO ALTO CA 94301

PLEASE MAKE SURE YOU'VE:
• Signed and dated the form
• Filled out the form completely
• Fax to: 1-650-321-3818 or fold, tape and mail (*please do not staple*)

# FREE Subscription
## To Qualified Applicants

# Introducing *Enterprise Development* magazine

ENTERPRISE DEVELOPMENT
Model Application:
The Rules Behind Cisco's Multibillion-Dollar Web Transactions
Data Architecture for E-commerce
Making NT's Middleware Scale
The Components of SAP Customization
An Application Architecture
For Rapidly Changing Business Rules
Battle of the Suites: IBM's Steve Mills
Make Your Apps Scale On NT Clusters
NT Development For Unixheads
Extending a Legacy Application to the Web

## In every issue:

- **Case studies that don't leave out the "gotchas"**
- **How to get more from programming teams....and your middleware**
- **How to integrate software with ERP systems such as SAP and Peoplesoft**
- **Achieving and documenting a ROI for improvements in the development process**

## *All the information you need to architect enterprise apps and better manage the development process*

Until now, no single magazine has met the needs of enterprise development managers. They've had to scour countless publications aimed at CIOs or coders in search of solutions to their specific problems.

Finally there's a magazine that cuts through all the confusion to bring the right answers together under one cover: *Enterprise Development.* **And it's free to qualified subscribers.***

*Enterprise Development* is written by and for IS professionals responsible for the design and management of application development projects. It's an independent source that will investigate and evaluate what to use in enterprise development projects. It examines the good, bad and ugly of new tools, technologies and techniques, and includes case studies on everything from n-tier MTS applications to simple extensions of legacy applications out through the Web. *Enterprise Development* is designed to help projects scale-up in terms of users supported, without also scaling-up the budget and time-to-market.

**Don't waste another minute–reserve your FREE SUBSCRIPTION today. Here's how:**

**Subscribe online at**

**www.enterprisedev.com**

*Publisher reserves the right to determine qualification for free subscriptions.

FTP FAWCETTE TECHNICAL PUBLICATIONS

209 Hamilton Avenue
Palo Alto, CA 94301-2500
Phone: 650.833-7100

Windows NT is a registered trademark of Microsoft Corporation.
PeopleSoft is a registered trademark of PeopleSoft Inc.
All other brand names or trademarks are the property of their respective companies.

when necessary is often referred to as "lazy instantiation." Unused classes can't be created, and created classes can't be duplicated, keeping the SortFactory's single instance as efficient as possible.

The test harness is often overlooked when implementing code. For our purposes, we'll use the Java 2 Integer class, which implements the Comparable interface. Our test elements will be integer values but any Comparable type works equally well.

Our SortingTest class in Listing 4 implements the harness. The `randomList()` method generates a List collection with the specified number of randomized elements. The `printList()` method is used to display our results in a readable format. We also need a `copyList()` method to do a deep copy. We'll need to use separate copies of the input Lists because our algorithms sort by reference. Finally, we implement a `sortedList()` method, which tests to see if a list is sorted. It does this by checking all adjacent element pairs and making sure they are in the right order.

The main clause in our test harness loops for several iterations. On each loop, a random sample is generated and each of the algorithms are tested. For each test, we copy the test sample, sort it, and display both the original list and the sorted list, along with a PASSED or FAILED message to show whether the results were properly sorted. If any of the tests fail, our success flag gets set to false and the full test result indicates a failure.

*Listing continued from previous page.*

```
      while (count < algorithm.length)
      {
        List copy = copyList(list);
        Sort sort = SortFactory.getInstance(
          algorithm[count]);
        String name = sort.getClass().getName();

        if(!(name.startsWith(algorithm[count])))
        {
          throw new IllegalArgumentException(
            "Algorithm failed" );
        }
        else
        {
          long mark = mark();
          sort.sort(copy);
          averages[j][count] += time(mark);
          System.out.print( "." );
          count++;
        }
      }
    }
```

```
    }
    System.out.println();
    System.out.print( "Elements: " );
    for(int i = 0; i < elements.length; i++)
    {
      if(i > 0) System.out.print( ", " );
      System.out.print( elements[i] );
    }
    System.out.println();
    for(int i = 0; i < algorithm.length; i++)
    {
      System.out.print((
        String.valueOf(algorithm[i])) + "Sort: ");
      for(int j = 0; j < elements.length; j++)
      {
        long average = averages[j][i] / (long)iterations;
        if(j > 0) System.out.print( ", " );
        System.out.print(average);
      }
      System.out.println();
    }
    System.out.println();
  }
}
```

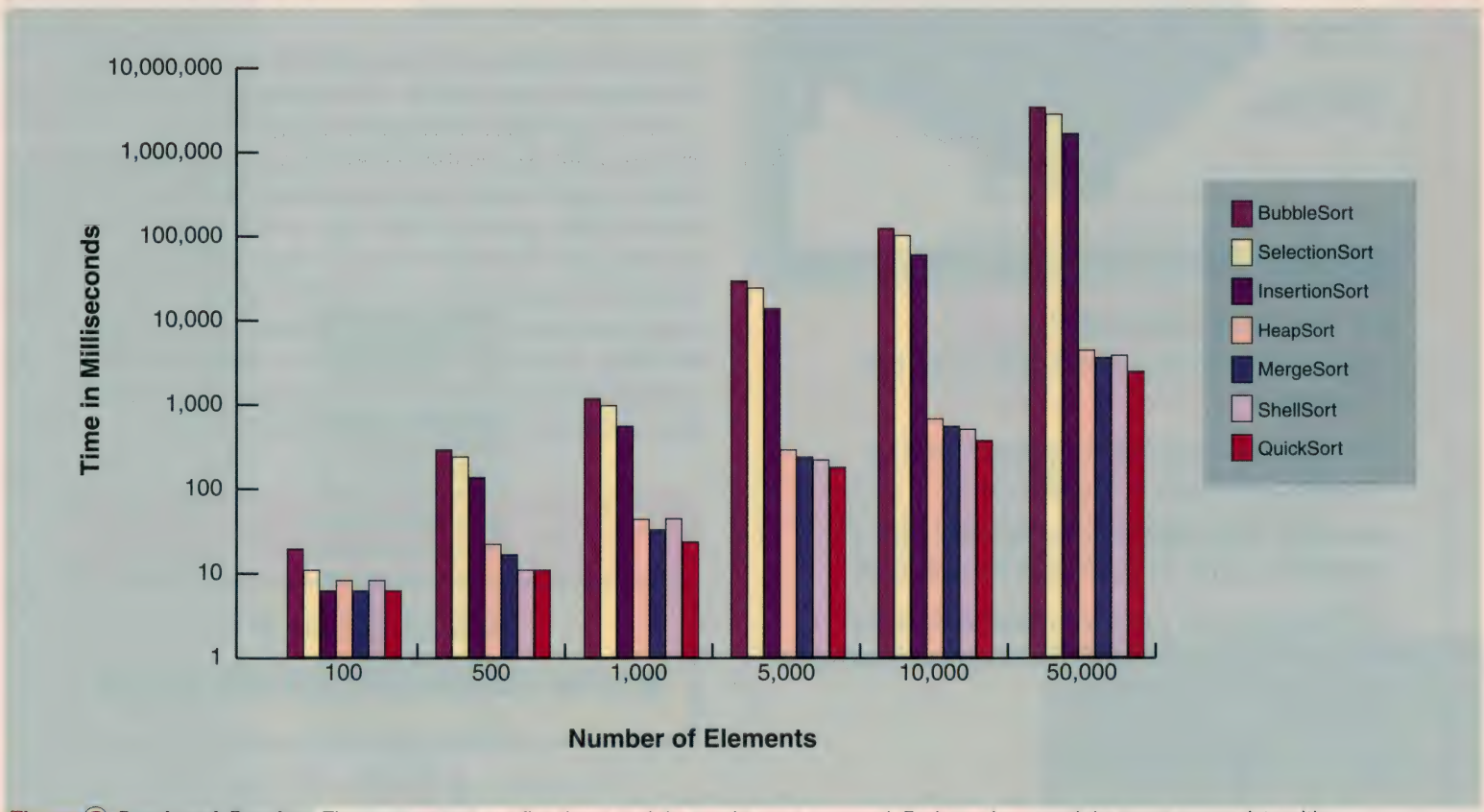



**Figure 2 Benchmark Results** The sort test was run five times, and the results were averaged. Each test loop used the same source data with each of the alogrithms.

This harness allows us to run large tests to certify code before delivery. This is an important technique used by quality assurance teams to do regression testing without manual intervention. It's useful to think of these as unit tests, implemented during development rather than as an afterthought. The net improvements in code quality, diminished debugging, and maintenance time are well worth the small investment you make up front.

### Preliminary Benchmarking

It's important to make intelligent decisions about which algorithm to apply to your application. A number of factors will come into play, but your decision will probably depend on the number of elements you are sorting and/or the level of order in the set to be sorted.

I performed a basic test based on random lists of a given number of elements (see Listing 5). To make this reasonably statistically relevant, several runs were made for each list size so that the resulting values are averages. However, there are a number of factors that are not accounted for, including memory constraints and type comparison (we used a simple Integer data type, and numbers are likely to vary for strings or compound data structures), nor were the algorithms optimized for speed.

Still, it is useful to be able to compare the algorithms, even if the comparison is coarse. Figure 2 shows the comparative results. Notice that the graph uses a logarithmic scale, so small differences are especially pronounced toward the top of the graph. It's also worth noting that

**It's important to make intelligent decisions about which algorithm to apply to your application**

the InsertionSort beat the others for small sets. QuickSort appears to beat out the others, but HeapSort consumes the least amount of memory, so it could be argued that it offers the best tradeoff. Some of these results may be attributable to the order of the random data, which is not strictly random but pseudorandom, as generated by the `Math.random()` method. Using the SortFactory, you can easily experiment with the type of data you use to optimize the algorithm for your circumstances.

The classes presented in this article provide an inventory of sorting algorithms, a uniform interface, and an efficient factory for generating these algorithms and using them in a variety of interchangeable circumstances. Applications can take advantage of algorithm-specific characteristics to optimize sorting speeds under a variety of data conditions. The use of interfaces and programming patterns in our implementation maximize reuse and extensibility, and the test harness guarantees reliability. ■

### Code Online

All the code in this article is available for free at www.java-pro.com. Premier Club members can also access additional source code and more information about O notation. Use the Locator+ codes shown below to download the code. Free online registration is required. See the Online Resources box in "Letters to the Editor" for more information.

**A SortFactory for the Collections API**
Registered Locator+: **JP0299**
Premier Locator+: **JP0299CD**

TechTIP

## Coming Events

If you're extending an AWT Component, such as a Canvas to write your own control, you'll probably want it to trigger events that the super class does not handle. It is not hard to add the necessary logic, so we'll use an `ActionEvent` as an example. First, your class will need somewhere to keep track of any number of Listeners that may be waiting for the event. There is an `ActionListener` class for this purpose:

```
import java.awt.event.*;
  private ActionListener list;
```

Then you need two methods so that other objects can add and remove themselves from this list:

```
  public synchronized void addActionListener(ActionListener al)
    {
    list = AWTEventMulticaster.add(list, al);
  }
  public synchronized void removeActionListener(ActionListener
    al) {
    list = AWTEventMulticaster.remove(list, al);
  }
```

Finally, at the point in your code where you wish to create an event, it needs to be sent to all waiting Listeners:

```
  if (list != null)
    list.actionPerformed(new ActionEvent(this,
    ActionEvent.ACTION_PERFORMED, yourCommand));
```

There are similar classes to ActionListener for all the other event types that you may need, in the java.awt.event package.

*—Tom Yarker*

## Which Browser is Running Me?

Although Java's ability to run on numerous computer platforms makes a developer's task easier, it is sometimes necessary to customize code to match the abilities of the environment. A good example is the need to tailor the generation of Dynamic HTML syntax to the brand of browser that will be used to display it. Surprisingly, there doesn't seem to be a direct technique for determining the name of the program that invoked the Java Virtual Machine. The following method can be inserted in any class and will return a value suitable for use in a switch:

```
// Gets run-time platform
// Returns 0 for an Application
//   1 for a Netscape  browser
//   2 for a Microsoft browser
//  -1 for an unknown platform
  public int getPlatform() {
    SecurityManager sm = System.getSecurityManager();
    if (sm == null) return 0;
    String st = sm.toString().toLowerCase();
    if (st.startsWith("netscape")) return 1;
    if (st.startsWith("com.ms." )) return 2;
    return -1;
  }
```

*—Tom Yarker*

# Introduction to the Datagram Protocol

*With Java, it's easy to support TCP and UDP servers, implement multiple servers of either type, and provide concurrent services to multiple clients*

by Richard G. Baldwin

The User Datagram Protocol (UDP) is an alternative to TCP for transmitting data through the Internet. Unlike TCP, UDP provides no guarantee that the datagram packets (self-contained, stateless entities) will arrive at their destination, or if they do arrive, that they will arrive in the correct order. However, packets sent via UDP arrive quicker than those sent with TCP. In some cases, such as real-time audio, arrival speed takes priority over an arrival guarantee. In this article, the last in our series on network programming, I will show you how to write UDP client and UDP server programs.

UDP programming differs from TCP programming in at least two respects: There is no ServerSocket concept for datagrams, and you must construct your own packets for UDP. With UDP, the DatagramSocket class is used by both the client and the server to construct the necessary sockets. The DatagramPacket class is used at both ends to construct and deconstruct packets of data that are sent and received via a DatagramSocket object.

To send data, construct a DatagramPacket object containing the required information, present it to a DatagramSocket object, and invoke the `send()` method on that object. To receive data, construct an empty packet, present it to a DatagramSocket object, and invoke the `receive()` method on that object. The thread will block on the `receive()` method until a packet of data is received via the network. That packet will then be used to populate your empty packet (or attempt to do so). All necessary information, such as addresses, port, and data, are contained in the packet.

A DatagramSocket listening on a specific port will receive packets from any client that happens to send a packet to that port. This can result in intermingled packets from different clients, and it is the responsibility of the application to sort them out. The way you sort out the packets will depend on the protocol you are implementing. The address of the sender can be extracted from the packet to assist in that sorting process.

Richard Baldwin is a Professor of CIS/CSC at Austin Community College in Austin, Texas, where he teaches introductory, intermediate, and advanced Java Programming courses. He also provides Java consulting and onsite training for various companies in central Texas. He publishes Java tutorials at www.phrantic.com/scoop/onjava.html. Reach him at baldwin@austin.cc.tx.us.

## DatagramPacket and DatagramSocket Classes

The DatagramPacket class provides two constructors: One for sending packets and one for receiving packets. Both constructors require you to provide a byte array along with the length of the array to construct a packet to receive data.

| Method | Return |
|---|---|
| `getAddress()` | Returns an `InetAddress` object. If the datagram was received from the network, the address represents the computer that sent the datagram. If the datagram was constructed locally, the address represents the computer to which it is to be sent. |
| `getPort()` | Returns the port from which the datagram was sent, or the port to which it is to be sent, depending on the origin of the datagram. |
| `getData()` | Returns a reference to a byte array containing the data portion of the datagram. You can use datagrams to communicate any kind of data that you can put into a byte array at one end and extract from a byte array at the other end. |
| `getLength()` | Returns the number of bytes contained in the data portion of the datagram. |

**Table 1 Extracting Information** These methods provide the means for extracting information from a datagram packet. The function of each method is reflected in its name.

**Listing 1 NetSockets07** This program adds UDP echo capability to the NetSockets05 program from last month's article. It performs two different echo tests with the same server by sending two lines of text to the standard echo port (7). The first test is a TCP/IP echo test, the second is a UDP echo test.

```
**********************************************************/
This is a TCP echo test
xxxxxxxxxxxxxxxxxxxxx
This is a UDP echo test
**********************************************************/

import java.net.*;
import java.io.*;
import java.util.*;

class NetSockets07{
  public static void main(String[] args){
    String server = "www.austin.cc.tx.us";
    int port = 7; //echo port
    String msg1 = "This is a TCP echo test";
    String msg2 = "This is a UDP echo test";

    //First conduct a TCP echo test
    try{
      //Get a socket, connected to the specified server
      //on the specified port.
      Socket socket = new Socket(server,port);

      //Get an input stream from the socket
      BufferedReader inputStream =
              new BufferedReader(new InputStreamReader(
                   socket.getInputStream()));

      //Get an output stream to the socket. Note
      //that this stream will autoflush.
      PrintWriter outputStream =
              new PrintWriter(new OutputStreamWriter(
                  socket.getOutputStream()),true);

      //Send line of text to the server
      outputStream.println(msg1);
      //Get echoed line back from server and display it
      System.out.println(inputStream.readLine());

      //Close the TCP socket
      socket.close();

      //Now conduct a datagram echo test to same port on
      //the same server

      //Convert the message to a byte array
      byte[] udpMsg = msg2.getBytes();
      InetAddress addr = InetAddress.getByName(server);
      //Create packet to send to the UDP echo port
      DatagramPacket packet =
        new DatagramPacket(udpMsg,udpMsg.length,addr,port);
      //Now get a datagram socket to send the message
      DatagramSocket datagramSocket = new DatagramSocket();
      //Now send the message
      datagramSocket.send(packet);

      //Now overwrite the msg in the packet to confirm that
      //echo is really received
      byte[] dataArray = packet.getData();
      for(int cnt = 0; cnt < packet.getLength(); cnt++)
        dataArray[cnt] = 'x';
      //Display overwritten version
      System.out.println(new String(packet.getData()));
      //Now receive the echo into the same packet. Echo
      //will overwrite current contents of the packet
      datagramSocket.receive(packet);
      //Display the echo
      System.out.println(new String(packet.getData()));
      datagramSocket.close();
    }//end try
    catch(UnknownHostException e){
      System.out.println(e);
      System.out.println(
              "Must be online to run properly.");
    }//end catch UnknownHostException

    catch(SocketException e){System.out.println(e);}
    catch(IOException e){System.out.println(e);}

  }//end main
}//end class NetSockets07
//=========================================================//
```

The data that is received will be deposited in your array. If the data won't fit into the array, the excess data will be lost and an `IllegalArgumentException` will be thrown.

When you construct a packet to send, you need to put the data into the array before invoking the `send()` method. The data portion of a datagram packet is limited to about 65,450 bytes. You also need to provide an `InetAddress` object containing the destination address and the port number at the destination. The address and port are stored in the packet object rather than being used to construct the socket as you would in TCP.

The methods of the DatagramPacket class provide the means for extracting information from a packet. The information provided by each method generally matches the name of the method (see Table 1). However, it is important to know how to interpret that information. For example, the `getAddress()` method returns an InetAddress object. If the datagram was received from the network, the address represents the computer that sent the datagram. If the datagram was constructed locally, the address represents the computer to which it is to be sent. The program must know which case applies in order to know how to interpret the address.

The DatagramSocket class, which is used both to send and receive a datagram, has two constructors: One allows you to specify the port and the other allows the system to pick an available port (anonymous). No matter which you use, the port from which the datagram is sent is placed in the datagram header and is available at the receiving end.

Typically, server software will use the constructor that specifies a predefined port. If not specified, client software wouldn't know to which port to address the datagram. Client software can use either constructor. Otherwise, there is no difference in the datagram sockets used by clients and servers.

**In some cases arrival speed takes priority over an arrival guarantee**

### Putting it Together

To recap, you specify the destination address and port when you construct the datagram packet. You send the datagram by invoking the `send()` method on an existing datagram socket and passing the packet object as a parameter to the `send()` method. When the packet is sent, the address and port number of the source computer are automatically placed in the header portion of the packet.

To listen for a datagram, instantiate a DatagramSocket object on a specific port. Then invoke the `receive()` method on that socket passing a reference to a packet object as a parameter to the `receive()` method. The `receive()` method

**Listing 2 NetServer02** This server supports echo processing of datagrams using UDP, providing us with a single program that supports three different servers at the same time on different threads.

```
import java.net.*;
import java.io.*;
import java.util.*;

public class NetServer02{
  public static void main(String[] argv){
    //Instantiate a new custom security manager.
    System.setSecurityManager(new ServerSecurityManager());
    //Instantiate a server object to listen on port 80
    HttpServer httpServerThread = new HttpServer();
    //Instantiate a TCP server object to listen on port 7
    EchoServer echoServerThread = new EchoServer();
    //Instantiate a UDP server object to listen on port 7
    UdpEchoServer udpEchoServerThread = new UdpEchoServer();
  }//end main
}//end class NetServer02
//=============================================//
//=============================================//

//This class is used to instantiate a security manager
// that confirms that files can be downloaded only from
// the current directory or a subdirectory of that
// directory. Note that other than checking for file
// downloading, this is a wide-open security manager. DO
// NOT install it on your network without considering the
// need to implement tighter security features.  This
// security manager class was tested using JDK 1.1.3
// under Win95.
class ServerSecurityManager extends SecurityManager{
  //This overridden method attempts to allow the server
  // to deliver files only from the current directory or
  // from a subdirectory of that directory.
  public void checkRead(String str){
    if(new File(str).isAbsolute()
                        || (str.indexOf("..") != -1))
      throw new SecurityException(
                  "Access to file: " + str + " denied.");
  }//end checkRead()

  //The following list of overridden methods causes the
  // security manager to be wide open and eliminates all
  // security checks other than the one provided by the
  // previous overridden method.
  public void checkAccept(String s, int i){}
  public void checkAccess(Thread t){}
  public void checkAccess(ThreadGroup g){}
  public void checkAwtEventQueueAccess(){}
  public void checkConnect(String s, int i){}
  public void checkConnect(String s, int i, Object o){}
  public void checkCreateClassLoader(){}
  public void checkDelete(String s){}
  public void checkExec(String s){}
  public void checkExit(int i){}
  public void checkLink(String s){}
  public void checkListen(int i){}
  public void checkMemberAccess(Class c, int i){}
  public void checkMulticast(InetAddress a){}
  public void checkMulticast(InetAddress a, byte b){}
  public void checkPackageAccess(String s){}
  public void checkPackageDefinition(String s){}
  public void checkPrintJobAccess(){}
  public void checkPropertiesAccess(){}
  public void checkPropertyAccess(String s){}
  public void checkRead(FileDescriptor f){}
//public void checkRead(String s){}//overridden above
  public void checkRead(String s, Object o){}
  public void checkSecurityAccess(String s){}
  public void checkSetFactory(){}
  public void checkSystemClipboardAccess(){}
  public boolean checkTopLevelWindow(Object o)
    {return true;}
  public void checkWrite(FileDescriptor f){}
  public void checkWrite(String s){}
}//end class ServerSecurityManager
//=============================================//
//=============================================//

//This class is used to instantiate a UDP server thread
// that listens on port 7 which is the echo port.
class UdpEchoServer extends Thread{

  UdpEchoServer(){//constructor
    start(); //start the thread and invoke the run() method
  }//end constructor
  //-------------------------//

  public void run(){
    try{
      //Instantiate a DatagramSocket on port 7 (echo port)
      DatagramSocket datagramSocket =
                                      new DatagramSocket(7);
      System.out.println(
                      "UDP Echo Server Listening on port 7");
      //Loop and listen to port 7.  If a call is
      // received, spawn a UdpEchoConnection thread to
      // deal with it.
      while(true){
        //This program is limited to echo string lengths
        // of 1024 bytes.
        DatagramPacket packet =
                    new DatagramPacket(new byte[1024],1024);
        //This statement blocks on the receive() method
        // and populates the packet if a call is received.
        // The populated packet is passed as a parameter
        // to a new thread that is spawned to handle the
        // client. Then this thread instantiates a new
        // empty packet and goes back to listening.
        datagramSocket.receive(packet);
        new UdpEchoConnection(packet);
      }//end while loop
    }catch(IOException e){System.out.println(e);}
  }//end run

}//end class UdpEchoServer
//=============================================//

//This class is used to spawn a thread to deal with a UDP
// call that is received on port 7 which is the echo
// port.
class UdpEchoConnection extends Thread{
  DatagramPacket packet;

  UdpEchoConnection(DatagramPacket packet){//constructor
    System.out.println("Received a call on port 7");
    this.packet = packet;
    //Operate at a priority that is below the threads
    // listening on the ports.
    setPriority( NORM_PRIORITY-1 );
    start();//start this thread and invoke the run method
  }//end constructor
  //-------------------------//

  public void run(){
    System.out.println("Running UDP thread for port 7");

    //Create a packet to echo based on the data in the
    // packet that was received as a parameter.
    DatagramPacket packetToSend = new DatagramPacket(
                                    packet.getData(),
                                    packet.getLength(),
                                    packet.getAddress(),
                                    packet.getPort());

    //Declare datagram socket outside of try block
    DatagramSocket datagramSocket = null;

    try{
      //Open a new datagram socket
```

Continued on page 40.

e-business

Get a **free Web Self-Service Starter Pack,** including product trial code, and find out more about creating hot self-service apps quickly and easily, at www.software.ibm.com/websoftware

## And for your next trick, make this line disappear.

Customers using the Web to get what they need for themselves. That's the idea behind Web Self-Service. But first, you've got to grab them with hot self-service apps. Enter Web Self-Service solutions from IBM, with great products like WebSphere™ Studio software. Because it's Java™-based, its servlet creation wizards eliminate tedious CGI script and let you quickly build dynamic, interactive Web apps. And you can create scripts for Java, JavaScript, Jscript, HTML, D-HTML and JavaServer pages with NetObjects ScriptBuilder tools. All while NetObjects Fusion lets you visually create, edit and manage your entire Web site. The result: you can instantly preview your work, smoothly add language elements and quickly navigate to embedded functions and objects. Once you've created your apps, WebSphere Application Server lets you host Java servlets on most Web servers, with built-in connectors to tap data and apps you're already using. And if you need to build collaborative workflow applications for your rapidly changing business processes, Lotus® Domino™ Application Server supports secure, highly interactive Web Self-Service solutions. So creating hot Web apps has never been faster or easier. Find out more about these superior products at www.software.ibm.com/websoftware

IBM

Solutions for a small planet™

IBM, WebSphere, the e-business logo and Solutions for a small planet are trademarks of International Business Machines Corporation in the United States and/or other countries. Lotus and Domino are trademarks of Lotus Development Corporation in the United States and/or other countries. Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States and/or other countries. Other company product and service names may be the trademarks or service marks of others. © 1998 IBM Corporation. All rights reserved.

Continued from page 38.

```
      datagramSocket = new DatagramSocket();
      //Use the new datagram socket to send the message
      // and close the socket.
      datagramSocket.send(packetToSend);
      datagramSocket.close();
      System.out.println("UDP Socket closed");
    }//end
    catch(UnknownHostException e){
      System.out.println(e);
      datagramSocket.close();
      System.out.println("UDP Socket closed");
    }//end catch UnknownHostException

    catch(SocketException e){
      System.out.println(e);
      datagramSocket.close();
      System.out.println("UDP Socket closed");
    }//end catch SocketException

    catch( IOException e){
      System.out.println( "I/O error " + e );
      datagramSocket.close();
      System.out.println("UDP Socket closed");
    }//end catch IOException
  }//end run method
}//end class UdpEchoConnection

//=========================================================//
//=========================================================//

//This class is used to instantiate a server thread that
// listens on port 7 which is the echo port.
class EchoServer extends Thread{

  EchoServer(){//constructor
    start(); //start the thread and invoke the run() method
  }//end constructor
  //-------------------------//

  public void run(){
    try{
    //Instantiate a serverSocket on port 7 (echo port)
    ServerSocket serverSocket = new ServerSocket(7);
    System.out.println(
                    "TCP Echo Server Listening on port 7");
    //Loop and listen to port 7.  If a call is
    // received, spawn an EchoConnection thread to
    // deal with it.
    while(true)
      //This statement blocks on the accept() method
      // and returns a socket if a call is received.
      // The socket is passed as a parameter to the
      // new thread that is spawned.
      new EchoConnection(serverSocket.accept());
    }catch(IOException e){System.out.println(e);}
  }//end run

}//end class EchoServer
//=========================================================//

//This class is used to spawn a thread to deal with a
// call that is received on port 7 which is the echo
// port.
class EchoConnection extends Thread{
  Socket socket;

  EchoConnection(Socket socket){//constructor
    System.out.println("Received a call on port 7");
    this.socket = socket;
    //Operate at a priority that is below the threads
    // listening on the ports.
    setPriority( NORM_PRIORITY-1 );
    start();//start this thread and invoke the run method
  }//end constructor
  //-------------------------//

  public void run(){
    System.out.println("Running thread for port 7");
    BufferedReader inputStream = null;
    PrintWriter outputStream = null;

    try{
      //Get an input stream from the socket
      inputStream =
                new BufferedReader(new InputStreamReader(
                                  socket.getInputStream()));
      //Get  an output stream to the socket.  Note
      // that this stream is supposed to autoflush.
      outputStream =
                  new PrintWriter(new OutputStreamWriter(
                          socket.getOutputStream()),true);

      //Get input from the browser and echo it to
      // the browser
      String input = inputStream.readLine();
      outputStream.println(input);
      System.out.println("Got Input: "+ input);
      socket.close();
      System.out.println("TCP Echo Socket closed");
    }//end
    catch( IOException e){
      System.out.println( "I/O error " + e );
      try{
        socket.close();
        System.out.println("TCP Echo Socket closed");
      }catch(IOException evt){System.out.println(evt);}
    }//end catch IOException
  }//end run method
}//end class EchoConnection
//=========================================================//
//=========================================================//

//This class is used to instantiate a server thread that
// listens on port 80 which is the http port.
class HttpServer extends Thread{

  HttpServer(){//constructor
    start(); //start the thread and invoke the run() method
  }//end constructor
  //-------------------------//

  public void run(){
    try{
    //Instantiate a serverSocket on port 80 (http port)
    ServerSocket serverSocket = new ServerSocket(80);
    System.out.println(
                    "TCP HTTP Server Listening on port 80");
    //Loop and listen to port 80.  If a call is
    // received, spawn an HttpConnection thread to
    // deal with it.
    while(true)
      //This statement blocks on the accept() method
      // and returns a socket if a call is received.
      // The socket is passed as a parameter to the
      // new thread that is spawned.
      new HttpConnection(serverSocket.accept());
    }catch(IOException e){System.out.println(e);}
  }//end run

}//end class HttpServer
//=========================================================//

//This class is used to spawn a thread to deal with a
// call that is received on port 80 which is the http
// port.
class HttpConnection extends Thread{
  Socket socket;
    BufferedReader inputStream = null;
    PrintWriter outputStream = null;
    DataOutputStream byteOutput = null;

  HttpConnection(Socket socket){//constructor
```

Continued next page 42.

blocks the thread until a datagram is received, so if you need to do anything else in the meantime (such as process a previously received packet), you should invoke the `receive()` method on its own thread. When the `receive()` method returns, your packet object will have been populated with the data in the datagram that was received.

When writing server software, you should bind the socket object to a predefined port number. When writing client software, you should listen on the same port that was used to send the initial datagram.

If you sent the datagram on an anonymous port, you can keep the socket open and use that same socket to listen for the reply. You can also invoke the `getLocalPort()` method on the socket before closing it to save the port number. Then you can open another socket on the same port. If you sent the initial datagram on a specified port, you can either use the same socket to listen for the reply, or you can close the original socket and get another socket on the same port.

To reply to an incoming datagram, extract the source address and port number from the incoming packet and place that information in a new packet along with the data to be sent in the reply. Then invoke the `send()` method on an existing DatagramSocket object and pass the packet object as a parameter. The address and port number of the sender will be automatically incorporated into the header of the packet when it is sent.

TCP and UDP port numbers aren't related. You can use the same port number for two processes if one communicates via TCP and the other communicates via UDP. It is common for servers to use the same port number and provide similar services using both protocols for some standard services (such as echo).

## UDP Echo Program

The NetSockets07 program (see Listing 1) adds UDP echo capability to the NetSockets05 program from last month's article ("Introduction to Socket Programming," January 1999). It performs two different echo tests with the same server by sending two lines of text to the standard echo port (7). The first test is a TCP/IP echo test, the second is a UDP echo test.

You must be logged onto an appropriate network for this program to run properly. If you connect to a server that supports both TCP and UDP echo testing, the output from this program should be:

```
This is a TCP echo test
xxxxxxxxxxxxxxxxxxxxxxx
This is a UDP echo test
```

In discussing the code in this program, I will skip the TCP code that was covered last month. I will also omit the exception handling code for brevity.

Typically, the program begins by defining the name of the server and the port number. The standard echo port number is 7 for both the TCP and UDP echo port protocols:

```
Class NetSockets07{
  public static void main(String[] args){
    String server =
                "www.austin.cc.tx.us";
    int port = 7; //echo port
```

Two String objects are instantiated, one to be used for the TCP echo test and the other to be used for the UDP echo test:

```
    String msg1 =
           "This is a TCP echo test";
    String msg2 =
           "This is a UDP echo test";
```

Next, convert the message to a byte array and instantiate an InetAddress object that identifies the server. Use the `getBytes()` method of the `String` class to convert the message to a byte array:

**When writing server software, you should bind the socket object to a predefined port number**

```
byte[] udpMsg = msg2.getBytes();
InetAddress addr = InetAddress.
                    getByName(server);
```

Instantiate a DatagramPacket object, providing the byte array and its length along with the address and port number of the server as parameters:

```
DatagramPacket packet =
    new DatagramPacket(
      udpMsg,udpMsg.length,addr,port);
```

Following this, instantiate an anonymous DatagramSocket object. The term anonymous could be confusing. The object is anonymous because the port number was not specified, not because it doesn't have a named reference variable:

```
DatagramSocket datagramSocket =
                    new DatagramSocket();
```

Use the DatagramSocket object to send the datagram to the server by invoking the `send()` method on the object. This causes the local address and port number to be incorporated into the packet and causes the packet to be sent to the address and port number that were encapsulated in the DatagramPacket object when it was constructed. The same DatagramSocket and DatagramPacket objects are used later to receive the packet that is (hopefully) sent back by the server:

```
datagramSocket.send(packet);
```

Because the same packet will be used to receive the echo, this code overwrites the data in the DatagramPacket object with "x." This makes it possible to demonstrate that the received data in the packet is new data and is not the residue of the message originally placed in the packet. This code also illustrates how you can access the data portion of a DatagramPacket object for other purposes:

```
byte[] dataArray = packet.getData();
for(int cnt = 0; cnt <
            packet.getLength(); cnt++)
  dataArray[cnt] = 'x';
```

Next, I display the overwritten data bytes, which simply displays a string of x characters:

```
System.out.println(
        new String(packet.getData()));
```

At this point, I assume there will be a reply from the server, so I invoke the `receive()` method on the same DatagramSocket object that was used to send the original message to the server. This causes the thread to block and wait for the reply. It is also possible to use the `setTimeout()` method to specify the length of time that the `receive()` method will block and wait:

```
DatagramSocket.receive(packet);
```

Once the packet is received, the thread is no longer blocked and program control moves to a statement that displays the data contained in the DatagramPacket object. Hopefully, the data is an echo of the message originally sent to the echo port on the server. Then the socket is closed and the program terminates:

```
System.out.println(
        new String(packet.getData()));
datagramSocket.close();
```

## UDP Server Program

I have also expanded the server program, NetServer01, from last month's article to support echo processing of datagrams using UDP. This gives us a single program that supports three different servers at the same time on different threads:

- Echo processing using UDP
- Echo processing using TCP
- HTTP protocol using TCP

This expanded version, NetServer02 (see Listing 2), is designed to support a UDP Datagram echo server on port 7. This server extracts the byte array contained in a datagram received on port 7 and sends that data back to the client where it originated. The cautions regarding server security provided in last month's article apply to this program also. In particular, because you will be connecting your computer to a network where it is vulnerable to intrusion, you should protect your data from unauthorized intrusion. A properly designed security manager can help to provide that protection.

The three different servers were implemented on different ports to illustrate the manner in which threads can be used in Java to service multiple ports. This program also illustrates that a single program can provide a mixture of TCP and UDP servers. Both the UDP echo portion of this program and the TCP echo portion can be tested using the program named NetSockets08 (see Listing 3), which was designed specifically for testing the echo portion of the program.

I will begin with the code that instantiates a security manager and three server thread objects. The security manager and the two TCP servers were covered in the earlier article and won't be discussed here. The statement that instantiates `UdpEchoServerThread` is new to this article. The three servers operate concurrently and asynchronously on different threads:

*Continued from page 40 .*

```
    System.out.println("Received a call on port 80");
    this.socket = socket;
    //Operate at a priority that is below the threads
    // listening on the ports.
    setPriority( NORM_PRIORITY-1 );
    start();//start this thread and invoke the run method
  }//end constructor
  //-------------------------//

  public void run(){
    System.out.println("Running thread for port 80");

    try{
      //Get an input stream from the socket
      inputStream =
                new BufferedReader(new InputStreamReader(
                                socket.getInputStream()));

      //Get  an output stream to the socket to use for
      // sending text strings.  Note that this stream
      // will autoflush.
      outputStream =
                 new PrintWriter(new OutputStreamWriter(
                         socket.getOutputStream()),true);

      //Get an output stream to the socket to use for
      // sending an array of byte data to a client.
      byteOutput = new DataOutputStream(
                                socket.getOutputStream());

      //Get request from the browser
      String request = inputStream.readLine();
      System.out.println("Got Request: "+ request);

      //Now analyze and attempt to respond to the request.
      // Note that this program supports only the GET
      // request.  Any other request will be ignored.
      StringTokenizer stringTokenizer =
                             new
                               StringTokenizer(request);
      if((stringTokenizer.countTokens() >= 2)
            && stringTokenizer.nextToken().equals("GET")){
        System.out.println("First token is GET");
        if((request = stringTokenizer.nextToken()).
                                        startsWith("/")){
          System.out.println(
                "Next token starts with /, strip it
                 off");
          request = request.substring(1);
        }//end if on startsWith("/")

        if(request.endsWith("/") || request.equals("")){
          System.out.println("Request endsWith / or " +
                             "blank, append
                               index.html");
          request = request + "index.html";
          System.out.println(
                           "Modified request: " +
                             request);
        }//end if on / or ""

        System.out.println("Try to read: " + request);
        FileInputStream fileInputStream =
                               new
                                 FileInputStream(request);
        //Instantiate a byte array whose size is equal to
        // the number of bytes that can be read from this
        // fileInputStream without blocking. Then read the
        // file into the byte array.
        byte[] data =
                     new
                       byte[fileInputStream.available()];
        fileInputStream.read(data);

        //Send the bytes in the array to the client.
        byteOutput.write(data);
        byteOutput.flush();

      }//end if stringTokenizer.countTokens...
      else
        //This happens when the request is not GET
        outputStream.println("<HTML><BODY><P>400 Bad " +
                             "Request<P></BODY></HTML>");

      socket.close();
      System.out.println("Socket closed");
    }//end
    catch(FileNotFoundException e){
      outputStream.println(
         "<HTML><BODY><P>404 Not Found<P></BODY></HTML>");
      try{
        socket.close();
        System.out.println("Socket closed");
      }catch(IOException evt){System.out.println(evt);}
    }//end catch

    catch(SecurityException e){
      System.out.println(e);
      e.printStackTrace();
      outputStream.println(
          "<HTML><BODY><P>403
            Forbidden<P></BODY></HTML>");
      outputStream.println(e);
      try{
        socket.close();
        System.out.println("Socket closed");
      }catch(IOException evt){System.out.println(evt);}
    }//end catch SecurityException

    catch( IOException e){
      System.out.println( "I/O error " + e );
      try{
        socket.close();
        System.out.println("Socket closed");
      }catch(IOException evt){System.out.println(evt);}
    }//end catch
  }//end run method
}//end class HttpConnection
//=============================================================//
```

Coming Fall 1999—A Conference for the Technology Elite

# Enterprise DEVELOPMENT 2000 CONFERENCE

Enterprise Computing's Peek Experience

Join acclaimed digital newsletter editor and publisher

## David Coursey

as he guides you through the maze of distributed computing technologies to only the best new stuff. Middleware, servers, modeling, applications—it's all here. See what's in store for the year 2000 when you get a look behind the curtain with FTP and David Coursey.

Coming next fall from an alliance of Fawcette Technical Publications and UPSIDE: **Enterprise Development 2000**, a conference for the technology elite. David Coursey, editor of the widely-quoted newsletter *coursey.com*, was recognized by Adweek's *Marketing Computers* magazine as the #1 newsletter editor in the computer industry three years running.* His insights and sometimes biting commentary on the computer industry have made him one of the most often quoted observers in the industry.

## Call today for more information

# 1-800-848-5523

or visit us online at
**www.enterprisedev.com**

PEOPLE • TECHNOLOGY • CAPITAL
UPSIDE
WWW.UPSIDE.COM

[c]
coursey.com

FTP FAWCETTE TECHNICAL PUBLICATIONS

209 Hamilton Avenue • Palo Alto, CA 94301-2500
Phone (650) 833-7100 • Fax (650) 853-0230
www.fawcette.com • www.devx.com

*from *Marketing Computers* June 1998, 1997 and May 1996 annual rankings of "Most Influential Computer Journalists"

Enterprise Development is a trademark of Fawcette Technical Publications.
UPSIDE magazine is a trademark of Upside Media Inc. and coursey.com is a trademark of David Coursey.

```
public class NetServer02{
  public static void main(String[] argv){
    System.setSecurityManager(
          new ServerSecurityManager());
    HttpServer httpServerThread =
                    new HttpServer();
    EchoServer echoServerThread =
                    new EchoServer();
    UdpEchoServer udpEchoServerThread =
                   new UdpEchoServer();
  }//end main
}//end class NetServer02
```

Whenever a server thread needs to service a client, it spawns another thread at a lower priority to service that client. Then the server thread goes back to listening for other clients that may be requesting service.

In this code the constructor for the class is used to instantiate an object to provide UDP echo services on port 7. This constructor simply invokes its own `start()` method to get the thread up and running. The `start()` method in turn invokes the `run()` method:

```
UdpEchoServer(){//constructor
    start();
    //start the thread and invoke
    // the run() method
}//end constructor
```

Now we'll examine the `run()` method of the UDP thread. For brevity, I omitted exception-handling code and some of the code that simply displays status while the server program is running. I begin by instantiating a DatagramSocket object and binding it to port 7:

```
DatagramSocket datagramSocket =
                  new DatagramSocket(7);
```

Next, I enter an infinite loop where I first instantiate an empty DatagramPacket object and then invoke the `receive()` method on the DatagramSocket object, passing the DatagramPacket object as a parameter. This program is limited to an incoming data length of 1024 bytes when the DatagramPacket object is instantiated. This value could be increased if necesary:

```
while(true){
  DatagramPacket packet =
            new DatagramPacket(
                  new byte[1024],1024);
  datagramSocket.receive(packet);
  new UdpEchoConnection(packet);
}//end while loop
```

The `receive()` method blocks the thread and waits for a datagram packet to arrive. When a packet arrives, it is used to populate the empty DatagramPacket object that was passed to the `receive()` method as a parameter. Then a new `UdpEchoConnection` thread is instantiated to handle the client request and the newly populated DatagramPacket object is passed as a parameter to the constructor for that thread.

Having disposed of the requirement to service the client, the `UdpEchoServer` thread goes back to the top of the loop, instantiates another empty DatagramPacket object, and blocks again waiting for the arrival of the next datagram packet.

That brings us to the `UdpEchoConnection` class from which the object is instantiated to service the client. All of the information available regarding this client is contained in the DatagramPacket object that the constructor receives as a parameter. This object is saved by the constructor for later use. The constructor sets the thread priority to a level below the level of the threads monitoring the ports so that the

**Listing ③ NetSockets08** You can use this code to test both the UDP echo and TCP echo portions of this program.

```
This is a TCP echo test
xxxxxxxxxxxxxxxxxxxxxx
This is a UDP echo test
*********************************************************/

import java.net.*;
import java.io.*;
import java.util.*;

class NetSockets08{
  public static void main(String[] args){
    String server = "localhost";
    int port = 7; //echo port
    String msg1 = "This is a TCP echo test";
    String msg2 = "This is a UDP echo test";

    //First conduct a TCP echo test
    try{
      //Get a socket, connected to the specified server
      //on the specified port.
      Socket socket = new Socket(server,port);

      //Get an input stream from the socket
      BufferedReader inputStream =
               new BufferedReader(new InputStreamReader(
                   socket.getInputStream()));

      //Get an output stream to the socket. Note
      //that this stream will autoflush.
      PrintWriter outputStream =
               new PrintWriter(new OutputStreamWriter(
                 socket.getOutputStream()),true);

      //Send line of text to the server
      outputStream.println(msg1);
      //Get echoed line back from server and display it
      System.out.println(inputStream.readLine());

      //Close the TCP socket
      socket.close();

      //Now conduct a datagram echo test to same port on
      //the same server

      //Convert the message to a byte array
      byte[] udpMsg = msg2.getBytes();
      InetAddress addr = InetAddress.getByName(server);
      //Create packet to send to the UDP echo port
      DatagramPacket packet =
        new DatagramPacket(udpMsg,udpMsg.length,addr,port);
      //Now get a datagram socket to send the message
      DatagramSocket datagramSocket = new DatagramSocket();
      //Now send the message
      datagramSocket.send(packet);

      //Now overwrite the msg in the packet to confirm that
      //echo is really received
      byte[] dataArray = packet.getData();
      for(int cnt = 0; cnt < packet.getLength(); cnt++)
        dataArray[cnt] = 'x';
      //Display overwritten version
      System.out.println(new String(packet.getData()));
      //Now receive the echo into the same packet. Echo
      //will overwrite current contents of the packet
      datagramSocket.receive(packet);
      //Display the echo
      System.out.println(new String(packet.getData()));
      datagramSocket.close();
    }//end try
    catch(UnknownHostException e){
      System.out.println(e);
      System.out.println(
                "Must be online to run properly.");
    }//end catch UnknownHostException

    catch(SocketException e){System.out.println(e);}
    catch(IOException e){System.out.println(e);}

  }//end main
}//end class NetSockets08
//=========================================================//
```

activity of the UdpEchoConnection thread cannot interfere with the ability of the other threads to properly service those ports:

```
class UdpEchoConnection extends Thread{
  DatagramPacket packet;

  //constructor
  UdpEchoConnection(
                DatagramPacket packet){
    this.packet = packet;
    setPriority( NORM_PRIORITY-1 );
    //start this thread and invoke the
    // run method
    start();
  }//end constructor
```

Once the datagram object is saved and the priority is adjusted, the constructor invokes the start() method which in turn invokes the run() method to get the thread up and running.

The only requirement of this thread is to send a copy of the data in the packet that was just received back to the client that sent it. The client's address and port number are already contained in the packet, having been placed there by the client software when the packet was sent.

The DatagramPacket object that was populated from that incoming packet can be sent back to the client as an echo without modification. However, to illustrate some of the other methods of the DatagramPacket class, I elected to use that object to construct a new object and send the new object back to the client. This is closer to what might be needed in a server with requirements more complex than simply to echo the incoming packet.

This code extracts information from one such object for the construction of a new object. It is simply one long statement that invokes a constructor of the DatagramPacket class, passing information obtained from the received packet as parameters to the constructor:

```
public void run(){
  DatagramPacket packetToSend =
                new DatagramPacket(
                  packet.getData(),
                  packet.getLength(),
                  packet.getAddress(),
                  packet.getPort());
```

The next step is to instantiate a new DatagramSocket object and use that object to send the new DatagramPacket object to the client by invoking the send() method on the socket object and passing the packet object as a parameter. Then, assuming that no exceptions have been thrown, the socket is closed and the thread terminates normally. If an exception has been thrown, there are several exception handlers provided to deal with the situation:

```
DatagramSocket datagramSocket = null;
datagramSocket = new DatagramSocket();
datagramSocket.send(packetToSend);
datagramSocket.close();
```

Using a variety of threads, you can mix TCP and UDP servers, implement servers of either type, and provide services to multiple clients concurrently. The multithreaded programming capability of Java along with the encapsulation of complex network programming issues into easily used classes make this possible. ■

## Code Online

All the code in this article is available for free at www.java-pro.com. Use the Locator+ codes shown below to download the code. Free online registration is required. See the Onlines Resources box in "Letters to the Editor" for more information.

***Introduction to the Datagram Protocol***
Registered Locator+: **JP0299**
Premier Locator+: **JP0299R2**

**Save 20% on the Best Java Books**

Buy any book and **save 20%** when you order through Marketplace on The Development Exchange. Not only do we have the **best Java books, we also carry the latest titles** on Web design and development, ActiveX, enterprise development and more! Plus you can get **40% off our Featured Books of the Week.** Select from our top titles for great savings—but act quickly because each week means another set of titles at incredible discounts!

**Order online at www.devx.com**
DEVX MarketPlace

**FTP Books, 209 Hamilton Ave., Palo Alto, CA 94301 • Tel: 650-833-7100 • Fax: 650-853-0230**

Java is a trademark of Sun Microsystems. ActiveX is a registered trademark of Microsoft Corporation.
The Development Exchange is a trademark of Fawcette Technical Publications.

# Dynamic Method Invocation with CORBA and Java

## *Reflection and Introspection enable you to create a highly configurable system*

by Luke Andrew Cassady-Dorion

In traditional applications the chain of method invocations is static—the method invoked is coded at compile time. This is true regardless of the implementation language. Of course, logical operators (such as if-then-else and switch) allow your program to make decisions, but the result of each decision is coded at compile time and cannot be changed at run time. In general, this is not a problem; however, certain applications require the ability to decide the chain of methods that are to be invoked dynamically at run time. Artificial Intelligence (AI) applications are the most common example of dynamic method invocation in action. AI applications "learn" from their experiences and, as a result, often "change their mind" about the desired action to take in a certain situation.

This article looks at developing dynamic invocation scripts using technologies provided by JDK 1.1 and CORBA 2.3 (for an introduction to CORBA see the sidebar "What is CORBA?"). We'll examine Java-only technologies and then CORBA-only technologies, concluding with a discussion of the pros and cons of each technology and how to use them to solve real business problems.

Luke Cassady-Dorion, a Sun Certified Java 1.1 Programmer, is a lead developer with Healtheon Corporation. He has deployed many enterprise-wide solutions written entirely in Java. Luke is the author of numerous articles on distributed computing and Java, and co-author of the recently published, *Java Distributed Objects*. He can be reached at luke@luke.org.

### Reflection in Java

Dynamic method invocation in Java is facilitated by a technology called "reflection," which provides access to information about the fields, methods and constructors of loaded classes. JDK 1.02 introduced minimal reflection capabilities through methods in the java.lang.Object class; however, the inclusion of the java.lang.reflect package in JDK 1.1 allows advanced programming. The java.lang.reflect package, detailed in Table 1, contains a collection of methods and interfaces used to interact with an object when a method is being dynamically invoked on it. In addition to these classes and interfaces, reflection is aided by a collection of methods present in the class java.lang.Class. As an object is instantiated in a Java environment, it is paired with a Class object that describes its run-time properties. Through this Class object and the entities present in the java.lang.reflect package, it is possible to discover information about an object and create a dynamic-method invocation script.

| Class/Interface name | Purpose |
|---|---|
| java.lang.Class | Automatically created by the JVM when an object is instantiated. Maintains references to all information available to describe the object. |
| java.lang.reflect.Member | Describes information about a method or field present in a class. This interface is implemented by other classes in this package. |
| java.lang.reflect.Constructor | Represents information about and provides access to a class' constructor. |
| java.lang.reflect.Field | Represents information about and provides access to a member variable present in an object. |
| java.lang.reflect.Method | Represents information about and provides access to a method present in an object. |
| java.lang.reflect.Modifier | A collection of static methods and variables used when examining class, member, or method modifiers (for example, static or final). |

**Table 1 The Java Reflection Model** This collection of methods and interfaces from the java.lang.reflect package is used to interact with an object when a method is being dynamically invoked on it. Introduced with the 1.1 JDK, this collection offers expanded reflection capabilites and enables advanced programming.

## What is CORBA?

CORBA is a technology that allows communication between applications written in virtually any language, running on any architecture. Most people react to this description with a certain amount of disbelief. Communication between multiple applications and architectures introduces issues that range from datatype mapping at the language level (for example, converting a Java String object to a C-style string) to byte-order issues at the hardware architecture level.

CORBA functions through a collection of technologies housed under what is called the Object Management Architecture (OMA). The OMA, in turn, consists of the Object Request Broker (ORB), a collection of CORBAservices, a collection of CORBAfacilities, and a collection of CORBAdomains. Although all components of the OMA are software specifications, their purpose is easily grouped into one of two categories. The ORB acts as piping that ties together all entities in the CORBA universe. CORBAservices, CORBAfacilities, and CORBAdomains are all collections of add-on software that provide application- or object-level functionality to the developer.

You will also need to be aware of a language called Interface Definition Language (IDL). Unlike programming languages such as Java and C++, IDL serves not to implement functionality, but to describe functionality. There are mappings from virtually every existing programming language to the IDL, allowing applications to describe their functionality in a manner that applications written in alternate languages can understand.

All CORBA development in this article uses the Inprise Visibroker for Java ORB. A free trial download is available at www.inprise.com. A full discussion of CORBA is beyond the scope of this article, but can be found in a variety of books including *Java Distributed Objects* (Sams 1998) and on the OMG Web site at www.omg.org.

—*L.A.C.D.*

With a basic understanding of reflection, we now dive into three examples that demonstrate this technology. Table 2 presents the methods in the Class class used during reflection. These steps use reflection to dynamically invoke methods on an object:

1. Obtain a reference to the object.
2. Obtain the object's run-time Class object by invoking the `getClass()` method inherited from the java.lang.Object parent class.
3. Obtain a reference to a method exposed by the Class object using the `getDeclaredMethod()` method (or related method from Table 2).
4. Invoke the method by calling its `invoke()` method.

In this code, a java.util.Vector object is manipulated using reflection:

```
Vector vec = new Vector();
```

| Method | Purpose |
|---|---|
| getConstructor(Class[]) | Obtains the Constructor object associated with the specified class. |
| getConstructors() | Obtains all Constructor objects associated with the target object. |
| getDeclaredClasses() | Obtains all Class objects associated with all fields in the target object. |
| getDeclaredConstructor(Class[]) | Obtains the Constructor object associated with the specified parameter. |
| getDeclaredConstructors() | Obtains all Constructor objects. |
| getDeclaredField(String) | Obtains the specified Field object. |
| getDeclaredFields() | Obtains an array of all available Field objects. |
| getDeclaredMethod(String, Class[]) | Obtains the specified Method object. |
| getDeclaredMethods() | Obtains all specified Method objects. |
| getField(String) | Obtains the specified Field object. |
| getFields() | Obtains all available Field objects. |
| getInterfaces() | Obtains the run-time Class objects associated with the interfaces implemented by the target object. |
| getMethod(String, Class[]) | Obtains a reference to the specified Method object. |
| getMethods() | Obtains references to all available methods. This includes Method objects inherited from a parent class. |
| getModifiers() | Obtains the modifiers (static, final, and so on) associated with the class. |
| isArray() | Returns true if the Class object is associated with an array. |

**Table 2 Reflection Class Methods** These methods are exposed by the Class class used in reflection. They can be used to dynamically invoke methods on an object.

```
Class vecClass = vec.getClass();
Method methodSize =
  vecClass.getDeclaredMethod("size",
  new Class[0]);
Object result = methodSize.invoke(vec,
  new Object[0]);
System.out.println(result);
```

Instead of simply invoking the `size()` method on the Vector object, the `getDeclaredMethod()` method is used to obtain a reference to the `size()` method and then tell it to invoke itself. When invoking `getDeclaredMethod()` to obtain the Method object associated with the `size()` method, two important parameters are passed. The first is a String object matching the signature of the target method, and the second is an array of Class objects matching the method's parameters. Because the `size()` method accepts zero parameters, an empty array is passed. Once the Method object is obtained, it is invoked using its `invoke()` method and is passed two parameters that specify the run-time course of action. The first parameter is the actual object on which the method is being invoked, and the second parameter is an array of Object objects associated with the parameters being passed to the method. Again, because `size()` accepts zero parameters, this array of Objects is empty.

At this point, you should understand reflection and how to perform a simple method invocation. We'll continue to

explore reflection by dynamically invoking a method that accepts object parameters and then invoking a method that accepts base-type parameters:

```
Vector vec = new Vector();
Class vecClass = vec.getClass();

// object parameters
String sFoo = new String("foo");
Object[] objParameters = {sFoo};
Class[] clsParameters = {
  new Object().getClass()};
Method methodAddElement =
  vecClass.getDeclaredMethod(
  "addElement", clsParameters);
methodAddElement.invoke(
  vec, objParameters);

// base type parameters
Integer[] objParameters = {new
  Integer(42)};
Class[] clsParameters =
  {objParameters[0].TYPE};
Method methodSetSize =
vecClass.getDeclaredMethod(
  "setSize", clsParameters);
methodSetSize.invoke(vec, objParameters);
```

In the original code example, calls to both `getDeclaredMethod()` and `invoke()` are passed empty arrays as their second parameter. In this example, we pass in populated arrays signifying that the method being dynamically invoked accepts parameters.

The first half of this example invokes the `addElement()` method, which accepts a single parameter of type java.lang.Object. To signify that the method accepts a single Object object as a parameter, the `getDeclaredMethod()` method is passed a single element array containing the run-time class associated with an Object object. In addition, the `invoke()` method is passed a single element array containing a String object to be inserted into the Vector.

**Fortunately, all base types in Java are paired with corresponding object wrappers**

In the second half of the new example, instead of invoking a method that accepts an object parameter, a method that accepts a base-type parameter is invoked. A tricky issue is raised because both the `getDeclaredMethod()` and `invoke()` methods accept arrays of objects indicating parameters to be passed into the target method and we are passing in base-type entities. Some encoding mechanism must be used that allows the base-type entities to masquerade as objects. Fortunately, all base types in Java are paired with corresponding object wrappers (int and Integer for example). When a base-type parameter is passed into a dynamically invoked method, the `invoke()` method is passed an array containing the base-type wrapper object, and the `getDeclaredMethod()` method is passed an array containing the Class object pointed to by the TYPE member variable exposed by the wrapper object.

To see these ideas in action, take a look at Listing 1. This code builds an applet that presents a color-changing Canvas object. The ob-

**Listing 1 The ColorChanging Applet** This applet presents a color-changing Canvas object.

```
import java.awt.*;
import java.applet.*;

public class ColorChangingApplet extends Applet {
   public void init() {
      ColorChangingCanvas colorChangingCanvas;
      colorChangingCanvas = new ColorChangingCanvas();

      InputPanel pnlInput;
      pnlInput = new InputPanel(colorChangingCanvas);

      setLayout(new BorderLayout());
      add(colorChangingCanvas, "Center");
      add(pnlInput, "South");
   }
}
import java.awt.*;
import java.awt.event.*;
import java.lang.reflect.*;

public class InputPanel extends Panel implements
ActionListener {
   private TextField     _txtSignature;
   private Button        _btnInvoke;
   private ColorChangingCanvas _colorChangingCanvas;

   public InputPanel(ColorChangingCanvas
      colorChangingCanvas) {
      _colorChangingCanvas = colorChangingCanvas;

      _txtSignature = new TextField(20);
      _btnInvoke = new Button("Invoke");

      setLayout(new FlowLayout());
      add(_txtSignature);
      add(_btnInvoke);

      _btnInvoke.addActionListener(this);
   }

   public void actionPerformed(ActionEvent ae) {
      String sSignature = _txtSignature.getText().trim();
      try{ Method methTarget =
         _colorChangingCanvas.getClass().
         getDeclaredMethod(sSignature,
         new Class[0]);
            methTarget.invoke(_colorChangingCanvas, new
            Object[0]);
      }
      catch( Exception e ) { System.out.println(e); }

   }
}
import java.awt.*;

public class ColorChangingCanvas extends Canvas {

   public ColorChangingCanvas() {
      setBackground(Color.red);
   }

   public void setToRed() {
      setBackground(Color.red);
      repaint();
   }

   public void setToBlack() {
      setBackground(Color.black);
      repaint();
   }

   public void setToYellow() {
      setBackground(Color.yellow);
      repaint();
   }
}
```

# Subscribe to Web Builder CD

## The Quarterly Web Design and Development CD-ROM

**4 WBCD issues only $29.95**

WEB BUILDER CD Quarterly Design & Development

**YES! Send me 4 issues of Web Builder CD for just $29.95**

Subscribe and save 25% off the newsstand price!

NAME (please print)

ADDRESS

CITY STATE ZIP

COUNTRY POSTAL CODE

E-MAIL

PHONE/FAX

**Payment enclosed** ☐ *Charge my:* **MasterCard** ☐ **AMEX** ☐ **VISA** ☐

**Or Order Online: www.devx.com**

**Card #** _____ **Exp. Date** _____

**Signature** _____

One year is four monthly issues. Subscription begins upon receipt of payment. Please allow four weeks for delivery of first issue. International subscriptions add $15 for shipping and handling. Payment must be made in U.S. dollars plus postage. Fawcette Technical Publications • 209 Hamilton Ave. • Palo Alto, CA 94301 • 650.833.7100 or 800.848.5523. For faster service, fax your order to 650.321.3818.

**WBJP02**


**4 WBCD issues only $29.95**

WEB BUILDER CD Quarterly Design & Development

**YES! Send me 4 issues of Web Builder CD for just $29.95**

Subscribe and save 25% off the newsstand price!

NAME (please print)

ADDRESS

CITY STATE ZIP

COUNTRY POSTAL CODE

E-MAIL

PHONE/FAX

**Payment enclosed** ☐ *Charge my:* **MasterCard** ☐ **AMEX** ☐ **VISA** ☐

**Or Order Online: www.devx.com**

**Card #** _____ **Exp. Date** _____

**Signature** _____

One year is four monthly issues. Subscription begins upon receipt of payment. Please allow four weeks for delivery of first issue. International subscriptions add $15 for shipping and handling. Payment must be made in U.S. dollars plus postage. Fawcette Technical Publications • 209 Hamilton Ave. • Palo Alto, CA 94301 • 650.833.7100 or 800.848.5523. For faster service, fax your order to 650.321.3818.

**WBJP02**

# Subscribe and Save 25% Off the Newsstand Price!

Everything you need to build a site, including development tools, sample apps and templates, reference materials and links, all on one CD-ROM.

**For Faster Service Call: 1-800-848-5523**

**or Subscribe Online at: www.devx.com**

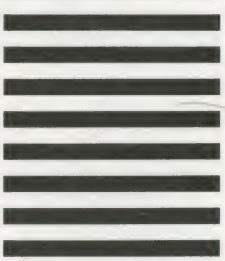
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

**BUSINESS REPLY MAIL**
FIRST-CLASS MAIL PERMIT NO. 317 PALO ALTO, CA

POSTAGE WILL BE PAID BY ADDRESSEE

FTP FAWCETTE TECHNICAL PUBLICATIONS
209 Hamilton Ave
Palo Alto, CA 94301-9796

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES



**BUSINESS REPLY MAIL**
FIRST-CLASS MAIL PERMIT NO. 317 PALO ALTO, CA

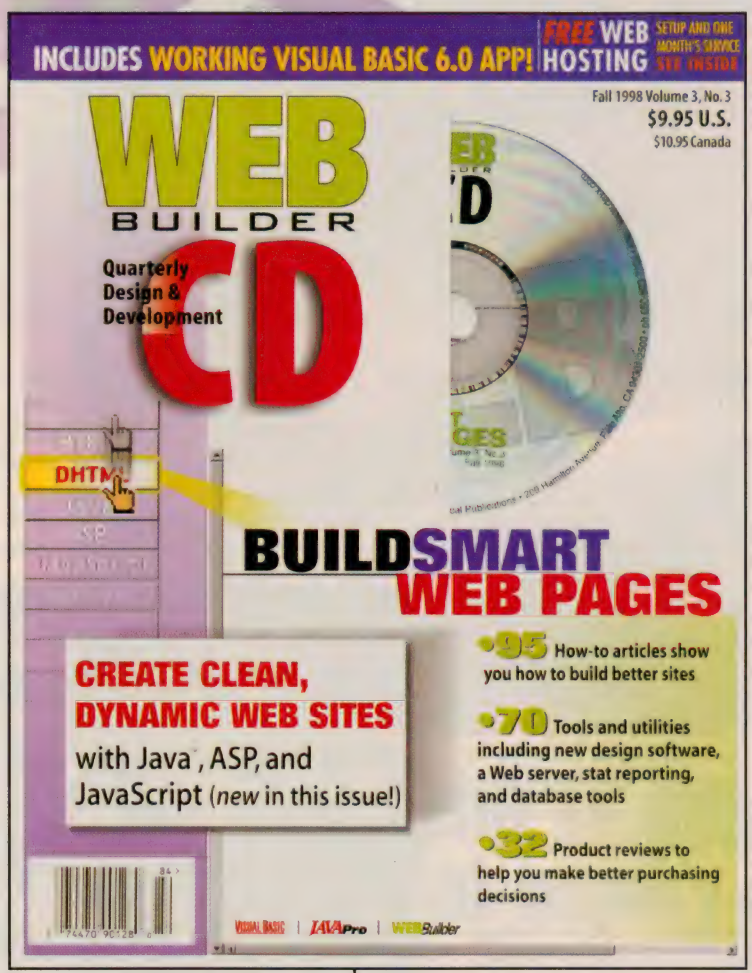
POSTAGE WILL BE PAID BY ADDRESSEE

FTP FAWCETTE TECHNICAL PUBLICATIONS
209 Hamilton Ave
Palo Alto, CA 94301-9796

# Build a better, smarter Web site

**Only $29.95 for four quarterly issues—you'll save 25% off the newstand price!**

## With *WEB Builder CD*, your complete Web Development toolkit


INCLUDES WORKING VISUAL BASIC 6.0 APP!
FREE WEB HOSTING
SETUP AND ONE MONTH'S SERVICE SEE INSIDE
Fall 1998 Volume 3, No. 3
$9.95 U.S.
$10.95 Canada
WEB BUILDER CD
Quarterly Design & Development
DHTML
BUILDSMART WEB PAGES
CREATE CLEAN, DYNAMIC WEB SITES
with Java, ASP, and JavaScript (new in this issue!)
•95 How-to articles show you how to build better sites
•70 Tools and utilities including new design software, a Web server, stat reporting, and database tools
•32 Product reviews to help you make better purchasing decisions
VISUAL BASIC | JAVAPro | WEBBuilder

All the Web development resources you need to take your site to the next level! Every issues delivers:

- Critical how-to articles from *VBPJ, Java™Pro* and *WEB Builder* magazines
- The latest versions of new tools and utilities
- Dozens of updated product reviews and fresh links

Plus, get valuable resources to continue and expand learning: links, software, tips, basics, code and more!

## Subscribe Today—4 issues for $29.95

Order online at **www.devx.com**
Or call **1-800-848-5523** International orders: 650-833-7100
Fax **650-321-3818**

FTP FAWCETTE TECHNICAL PUBLICATIONS **209 Hamilton Ave., Palo Alto, CA 94301**

Visual Basic is a registered trademark of Microsoft Corporation. Java is a trademark of Sun Microsystems. *VBPJ* is a trademark of Fawcette Technical Publications.

**Figure 1 Get Your Colors Done** To dynamically invoke a method, enter either "setToRed," "setToBlack," or "setToYellow" and click the "Invoke" button.

ject exposes a collection of methods that alter its background color, each of which is dynamically invoked when the user types in the name of the target signature and clicks the "Invoke" button. The applet is shown running in Figure 1.

## Introspection in CORBA

Dynamic method invocation in CORBA is facilitated by a technology similar to Java reflection that is called "introspection." As discussed in the sidebar, CORBA development in this article makes use of the Inprise Visibroker Object Request Broker (ORB) (www.inprise.com). If you do not have it installed on your system you'll need to download the trial version to run the example applications.

Introspection in CORBA is made possible by two complimentary technologies—the Interface Repository (IR) and Dynamic Interface Invocation (DII). The IR is a piece of software that contains the interfaces exposed by all active CORBA entities. Clients bind to the IR and query it to obtain descriptive information about existing CORBA entities. DII is the umbrella term covering all interfaces used when dynamically invoking operations on objects. Our discussion of CORBA introspection will start with interfacing with the IR and then move on to using the IR with DII.

The principle interface exposed by the IR is called the Repository and is supported by another collection of interfaces that describe different CORBA entities. These supportive entities are all named using the pattern `xxxDef`, where `xxx` is replaced with the entity being described. For example, an object that implements the `StructDef` interface describes a struct; see Table 3 for additional descriptive entities. In this code the IDL describes a stock including its last price and the symbol used to represent it:

```
struct StockS {
    string sSymbol;
    long iLastPrice;
};
```

The code in Listing 2 interfaces with the IR to obtain information on this struct. Next, start the Inprise ORB by typing `osagent` at a command prompt and type `irep MyIR` ("MyIR" can be replaced with any logical name given to the IR) to start the IR, and load your IDL file by choosing "Load..." from the "File" menu. Now compile the `StructPeek` class and execute the application by typing `java StructPeek` at a command prompt.

Once instantiated, a `StructPeek` object first binds to the ORB and then to the IR through its Repository interface. The `lookup()` operation exposed by the Repository interface is then invoked and passed the logical name of the entity that we desire additional information on. This operation is declared to return an abstract parent object because one of the different `xxxDef` objects might actually be returned. We know that we are looking up a struct, so we narrow (cast) the return value into a `StructDef` object. The `StructDef` object is then queried and the datatypes of its member attributes are displayed.

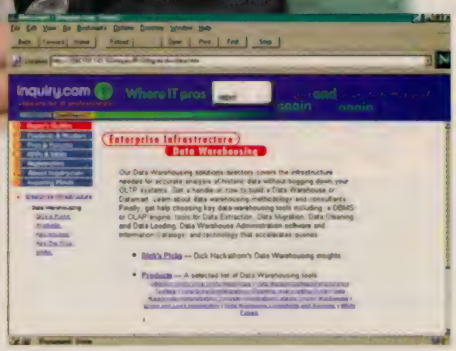
Once information about a particular CORBA entity has been discovered, it is possible to use DII to invoke operations on these entities. The primary entity encountered when performing DII, is the Request interface. This interface, detailed in Table 4, exposes operations that allow it to be tied to a target operation, have parameter values and modes specified, and a collection of operations that allow the target operation to be invoked. In the next example application,

**Listing 2 The StructPeek Application** By interfacing with the IR we can obtain information on StructDef.

```
import org.omg.CORBA.*;
import org.omg.CORBA.InterfaceDefPackage.*;

public class StructPeek {
     public StructPeek() {
     // bind to the ORB
     ORB orb = ORB.init();
     // bind to the Repository
     Repository repository = RepositoryHelper.bind(orb);

     // obtain the desription of the StockS struct
     StructDef structDef =
       StructDefHelper.narrow(repository.lookup("StockS"));

          // obtain references to all members
          StructMember[] members = structDef.members();

          // print details on each member
          for(int i=0;i<members.length; i++) {
               System.out.println("Member "+(i+1)+" Type: "+members[i].type);
               System.out.println("Member "+(i+1)+" Kind: "+
                                  members[i].type.kind());
          }
     }

   public static void main(String[] args) {
     new StructPeek();
   }
}
```

| CORBA entity | Descriptive Interface |
|---|---|
| alias | AliasDef |
| array | ArrayDef |
| attribute | AttributeDef |
| constant | ConstantDef |
| enum | EnumDef |
| estensible struct (Inprise specific) | EstructDef |
| exception | ExceptionDef |
| interface | InterfaceDef |
| module | ModuleDef |
| operation | OperationDef |
| parameter | ParameterDescription/ ParameterMode |
| string | StringDef |
| struct | StructDef |
| typedef | TypedefDef |
| union | UnionDef |

**Table 3 CORBA Entities** These interfaces are used to describe CORBA entities and support the Interface Repository. The Interface Respository and Dynamic Interface Invocation enable introspection in CORBA.




# "We found the IT answers we needed at inquiry.com."

When you need answers on development tools and techniques, turn to www.inquiry.com.

Our editors and "Ask the Pro" professionals provide unique insights on real-world development problems.

Now that inquiry.com is part of The Development Exchange suite of Web sites, you also have access to the largest database of independent product reviews, technical tips, and our Buyers Guide with information on over 5,000 tools for professional developers.

Nothing matches the efficiency and scope of www.inquiry.com.

Get the IT answers you need, free, today.

**Find us at www.inquiry.com**

PC MAGAZINE MAY 97 THE TOP 100 WEB SITES

inquiry.com
answers for IT professionals

© Inquiry.com, the inquiry.com logo, and The Development Exchange are trademarks of Fawcette Technical Publications.

| Operation | Description |
|---|---|
| any add_in_arg() | Adds an argument with the in mode. |
| any add_named_in_arg(in string name) | Adds the named argument that has the in mode. |
| any add_inout_arg() | Adds an argument with the inout mode. |
| any add_named_inout_arg(in string name) | Adds the named argument with the inout mode. |
| any add_out_arg() | Adds an argument with the out mode. |
| any add_named_out_arg(in string name) | Adds the named argument with the out mode. |
| void set_return_type(in ::CORBA::TypeCode tc) | Sets the return type for the operation invocation. |
| any return_value() | Obtains the return value, or null if the operations has yet to be invoked. |
| void invoke() | Invokes the target operation and waits for a return value. |
| void send_oneway() | Invokes the target operation and ignores the potential return value. |
| void send_deferred() | Invokes the target operation and immediately returns, the return value (if any) is obtained using the return_value() operation. |
| void get_response() | Blocks until the return value has been calculated by the server. |
| boolean poll_response() | Returns true if a return value is ready, false otherwise. |

**Table 4 Operations in the Request Interface** The Request Interface exposes operations that allow it to be tied to a target operation, have parameter values and modes specified, and a collection of operations that allow the target operation to be invoked.

we'll bind to an existing object, query the IR for that object's interface, and invoke an operation on it. Like the earlier struct, this interface describes a stock including its symbol and price:

```
interface StockI {    string
  getSymbol();    long getPrice();};
```

The interface is implemented in Listing 3. Note that the `main()` method includes the code used to interact with the ORB and Basic Object Adapter (BOA).

To get a Stock object up and running in the CORBA environment ensure that the ORB software is still running, compile the Stock class and StockI IDL, and enter the command `java Stock`. (If you are new to the Inprise environment, IDL is compiled using the application idl2java. Assuming the IDL is saved in a file titled "StockI.idl," compile it by entering `idl2java StockI.idl`.)

The code in Listing 4 binds to the Stock object, invokes its `_get_interface()` operation to obtain descriptive information about the interface, and uses DII along with the interface and operation descriptions to dynamically invoke operations. In this example, the method being dynamically invoked is called using the `invoke()` operation exposed by the Request interface. The `invoke()` method blocks until the return value has been calculated. If the return value is going to take some time to calculate, or if client does not need to know when the operation returns, the dynamic operation may be invoked in an alternate fashion. When the `send_oneway()` method is invoked, it immediately returns, and the operation is executed in an independent thread; the client has no knowledge when or if the remote operation finished. The `send_deferred()` operation also immediately returns, but the ORB waits for the return value in an independent thread. Once the ORB has obtained the remote operation's return value, it may be obtained by invoking the `return_value()` operation.

## A Few Caveats

Dynamic invocation of methods (using Java reflection or CORBA DII) enables you to create a highly configurable system. However, as the earlier examples demonstrated, there is additional overhead involved. The vast amount of code required to execute a method dynamically when compared to the simple one line required when performing a static invocation is enough to scare many developers away.

**Introspection in CORBA is made possible by two complimentary technologies**

In addition, the execution time required to create and invoke a method dynamically is much greater when compared to its static counterpart. Don't trade too much performance to make the application more extensible. Dynamic method invocation is also error prone. When statically invoking a method, the compiler performs a check to see if the method actually exists. In a dynamic environment, this compile time checking is not possible because the method being invoked is often not determined until run time. For example, in a static environment if you invoke the `size()` method exposed by the Vector class and misspell the method signature, the compiler would flag this as an error. However, in a dynamic environment, the compiler would not be able to sense the problem and an exception would be thrown at run time.

Keep these issues in mind as you begin to develop applications that could use either static

**Listing 3 The Stock Class** The main() method of the Stock class includes the code used to interact with the ORB and Basic Object Adapter (BOA).

```
import org.omg.CORBA.*;

public class Stock extends _StockIImplBase {

   public Stock() {
      super("MSFT");
   }

   public String getSymbol() {
      return "MSFT";
   }

   public int getPrice() {
      return 1013;
   }

   public static void main(String[] args) {
      // bind to the ORB and BOA
      ORB orb = ORB.init();
      BOA boa = orb.BOA_init();

      // create a new Stock object
      StockI msft = new Stock();
      // register the Stock object with the ORB
      boa.obj_is_ready(msft);
      System.out.println(msft+" is ready");

      // wait for incoming connections
      boa.impl_is_ready();
   }
}
```

or dynamic invocation of methods. Spend some time coding and examining both solutions.

### Solving Business Problems with Dynamic Invocation

DII is rather interesting because it introduces a new way of thinking to programmers, but we've yet to discuss how to use it to solve business problems. Developers love to play with a new technology and are often accused of implementing a slow or expensive solution just because it is new and fun to play with. Fortunately, it's fun to design an application using DII *and* it helps to solve many business problems.

As stated in the beginning of this article, AI applications are valid contenders for dynamic method invocation. Applications with complex decisions or detailed User Interfaces (UI) can also be added to this list.

Although Java no longer suffers from glacial execution times and is now suitable for enterprise development, applets still suffer some major problems. Downloading an applet often takes too long, and browser inconsistencies make UI development a difficult (if not impossible) process. For this reason, thin-clients are often best developed using HTML instead of Java. Because HTML cannot directly invoke methods on server objects, problems arise when HTML is used as a client language. A solution to this problem is to embed method invocations in an HTML form as arguments, parse out those arguments at the server, and use them to dynamically invoke the associated methods. This solution is too involved describe here, but an implementation of it is available online to Premier Club members of the Development Exchange. See the Code Online box for more details.

**It's fun to design an application using DII**

**Listing ④ The DynamicInvoker Class** The method being dynamically invoked in this example is called using the `invoke()` operation exposed by the Request interface.

```
import org.omg.CORBA.*;
import org.omg.CORBA.InterfaceDefPackage.*;

public class DynamicInvoker {

   public DynamicInvoker() {
      // bind to the ORB
      ORB orb = ORB.init();
      // bind to the Stock object
      StockI stock = StockIHelper.bind(orb);

      Repository repository = RepositoryHelper.bind(orb);

      // invoke its operations
      invokeAllOperations(stock);
   }


   private void invokeAllOperations(org.omg.CORBA.Object targetObject) {
      // obtain the interface description from the target object
      InterfaceDef interfaceDef = targetObject._get_interface();
      FullInterfaceDescription desc = interfaceDef.describe_interface();

      // obtain descriptions of all operations
      OperationDescription[] oppDefs = desc.operations;
      int iOppCount = oppDefs.length;
      for(int i=0; i<iOppCount; i++) {
         OperationDescription descActive = oppDefs[i];
         Request request = targetObject._request(descActive.name);

         // set the return type
         request.set_return_type(descActive.result);
         // invoke the operation
         request.invoke();
         // display the results
         System.out.println(request.result().value());
      }
   }

   public static void main(String[] args) {
      new DynamicInvoker();
   }
}
```

### Code Online

All the code in this article is available for free at www.java-pro.com. Augmented material, including the HTML demonstration, is available to Premier Club members. Use the Locator+ codes shown below to download the code. Free online registration is required. See the Online Resources box in "Letters to the Editor" for more information.

**Dynamic Method Invocation with CORBA and Java**
Registered Locator+: **JP0299**
Premier Locator+: **JP0299LD**

## TechTIP

### Create Rectangular Buttons

When placing JButtons in the JToolBar, use gif icons that are 25×25 pixels. Then create a ToolButton class that sets the insets to zero so that the buttons are square instead of rectangular:

```
public class ToolButton extends JButton
{

  public kmfToolButton(Icon img)   {
    super(img);
    setMargin(new Insets(0,0,0,0));
  }
```

*—James W. Cooper*

### Text Display

One nice feature of programs like Netscape's news reader is the ability to display the entire title of an article even though only part of it fits in the list box width. You can do the same thing by using setTooltipText to set the text to the complete list box entry each time the user clicks on a new line in the list.

*—James W. Cooper*

### Customize Buttons

JToggleButtons look good inside the JToolBar when you want buttons that stay down or up. You can also make them behave like radio buttons (only one down at a time) by adding them to a ButtonGroup. However, if you want to raise a depressed button when a operation is finished, you will need to write your own Mediator class instead of using the Mediator provided by the ButtonGroup.

*—James W. Cooper*

# Are We There Yet?

## JDK 1.2 contains over 1,600 APIs. Does it need any more changes?

by Daniel Savarese

On December, Sun gave Java developers an early Christmas present with the release of Java 1.2 (now called the Java 2 Platform). I expect that this version, like all the others, will be followed by the usual series of incremental releases. That means now is a good time to get a handle on what's new in Java 2, what's different from 1.1, what's missing, and to try to answer the question, "Are we there yet?"

Java 2 offers several new APIs. The sorely needed collections API in the `java.util` package provides a standard set of interfaces for implementing containers and iterating through their contents. The Java 2D API, found in several `java.awt` packages, including `java.awt.image`, `java.awt.font`, `java.awt.colors`, `java.awt.geom`, and `java.awt.print`, makes writing custom imaging applications and printing graphics a lot less painful. Swing and the accessibility API packages have been renamed to use `javax.swing` and `javax.accessibility` prefixes, much to the relief of many Java developers. A couple varieties of weak references have been added in the new `java.ref` package, allowing you to better manage memory use by interacting with the garbage collector. Many smaller changes have been added through new classes and methods that are scattered throughout the existing packages. For example, explicit thread local variable support is implemented by `java.lang.ThreadLocal` and temporary file creation is supported via the new `createTempFile` method in `java.io.File`. Veteran 1.1 programmers have a lot of new APIs to learn!

All right, so the API classes have evolved quite a bit. What about the virtual machine and the Java language itself? You'll find that the bytecode for the virtual machine has changed. If you want to generate class files that are downward compatible with 1.1, you have to pass the flag `-target 1.1` to the Sun JDK compiler. All Java 1.1 byte code is upward compatible with Java 2, although the APIs it depends on might have changed, requiring recompilation. If you obfuscate your code, you might find that it does not work with Java 2 VMs. The verifier in the 1.2 JDK complies much more closely to the JVM specification than the 1.1 JDKs. As a result, illegal bytecode that used to work could fail. As far as the language itself, there appear to be no real changes. We still don't have a preprocessor, templates of any kind, or even a limited form of operator overloading. These features might eventually find their way into the language—but not with the 1.2 release. However, there has been a change in floating point behavior. If you're not familiar with floating point, Java was designed to yield exactly the same results regardless of the platform you were running on. That means it would not take advantage of extra precision available on

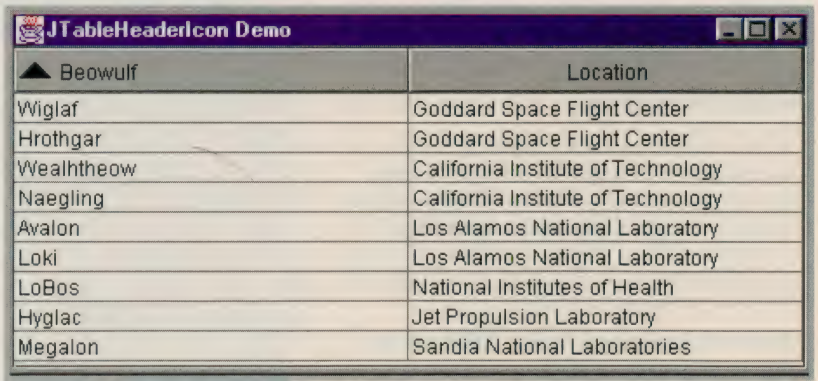



**Figure 1 Order Indicator** Each time the column header is clicked, the setIcon() method alternates between displaying the up arrow and the down arrow.

Daniel Savarese is a Senior Scientist at the Center for Advanced Computing Research at the California Institute of Technology. Daniel is the author of several Java packages, including the OROMatcher and PerlTools regular expression libraries, and the NetComponents network protocol library. He can be reached by e-mail at dfs@cacr.caltech.edu.

**Listing 1 Rendering Table Headers** You can draw icons and react to mouse events in JTable column headers by doing a little work with a table's TableColumnModel, accessible with getColumnModel().

```
import com.sun.java.swing.*;
import com.sun.java.swing.table.*;
import java.awt.*;
import java.awt.event.*;

class HeaderRenderer extends DefaultTableCellRenderer {
  private int __column;
  private TableCellRenderer __defaultRenderer;

  HeaderRenderer(TableColumn column) {
    __defaultRenderer = column.getHeaderRenderer();
    __column = column.getModelIndex();
  }

  public Component getTableCellRendererComponent(
     JTable table, Object value, boolean isSelected,
     boolean hasFocus, int row, int column){
    if(column == __column)
      return this;
    return __defaultRenderer.getTableCellRendererComponent(
     table, value, isSelected, hasFocus, row, column);
  }
}


public final class JTableHeaderIcon extends JFrame {
  private static final String[] __COLUMNS = {
    "Beowulf", "Location"
  };
  private static final String[][] __DATA = {
    { "Wiglaf", "Goddard Space Flight Center" },
    { "Hrothgar", "Goddard Space Flight Center" },
    { "Wealhtheow", "California Institute of Technology" },
    { "Naegling", "California Institute of Technology" },
    { "Avalon", "Los Alamos National Laboratory" },
    { "Loki", "Los Alamos National Laboratory" },
    { "LoBos", "National Institutes of Health" },
    { "Hyglac", "Jet Propulsion Laboratory" },
    { "Megalon", "Sandia National Laboratories" }
  };

  private JTable __table;
  private JScrollPane __scrollPane;
  private Icon __upIcon, __downIcon;


  public JTableHeaderIcon() {
    super("JTableHeaderIcon Demo");
    TableColumnModel model;
    TableColumn column;
    JTableHeader header;
    MouseAdapter headerListener;
    TableCellRenderer renderer;
    final HeaderRenderer label;
    Component component;

    __table = new JTable(__DATA, __COLUMNS);
```

Continued on next page.

some architectures. This has caused some scientists a lot of headaches and was addressed by Sun through a public review process announced in March 1998 (http://java.sun.com/pr/1998/03/pr980323.html). That review process led to the addition of a new keyword, called `strictfp`, that Sun slipped into Java 2. The new keyword is used to indicate that traditional Java floating point behavior should used (that of versions prior to 1.1). But the default floating point behavior has been changed so that it takes advantage of extra precision that may be available on the native platform. Watch out. Your floating-point code may now produce different (possibly more correct) results than in the past.

You might not have to learn how to use new language features, but you will have to learn the differences in the JDK tools. The Sun JDK no longer relies solely on your `CLASSPATH` environment variable to look for class files. It first looks in the bootstrap classpath, defined by the `sun.boot.class.path` system property. If it can't find a class there, it looks in the extension directories, defined by the `java.ext.dirs` system property. Finally, it will look in the application classpath, defined by the `java.class.path` system property. The Java Virtual Machine Debugger Interface and the Java Virtual Machine Profiler Interface are new additions that provide a means for extracting debugging and performance information from your programs. They are low-level interfaces that will likely be superceded by higher-level interfaces like the Java Debug Interface. My favorite tool change is the more-flexible documentation-generation facility. Javadoc now supports the notion of a "doclet," which is a program written with the provided doclet API that will customize your documentation output. You are no longer bound by the shackles of postprocessing default javadoc output with Perl scripts, or forced to rely on nonstandard documentation generators.

I'm a little overwhelmed by the scope of Java 2, mostly because there are so many darned classes. The days when you could program in Java without ever looking at the documentation are gone. Although a rich set of standard APIs are essential to establishing Java as a preferred development platform, the support for proven language constructs is vital as well. I am dismayed that templates, operator overloading, and preprocessing have gotten little or no attention from Sun in this release. I often meet scientists who are moving to Java and slamming headfirst into two obstacles—poor performance and the lack of operator overloading. A surprising number of scientists are willing to sacrifice performance in favor of portability, ease of development, and ease of maintenance, but without operator overloading it is cumbersome to write numerically intensive applications. Maybe this will change with the next major version, but until these and other lingering problems are addressed, the Java journey won't be over.

**Now is a good time to get a handle on what's new in Java 2**

**Q** How do I set the socket input and output buffer sizes in Java?

**A** You cannot set the socket input and output buffer sizes in Java versions prior to Java 2 without writing native code. You have to live with the system default values. However, with Java 2, `java.net.Socket` has acquired four new methods: `void setSendBufferSize(int size)`, `intgetSendBufferSize()`, `void setReceive-BufferSize(int size)`, and `int getReceiveBufferSize()`. You can use these methods to set

Continued from the previous page.

```
    __scrollPane = new JScrollPane(__table);
    setContentPane(__scrollPane);

    __upIcon   = new ImageIcon("up.gif");
    __downIcon = new ImageIcon("down.gif");

    model  = __table.getColumnModel();
    column = model.getColumn(0);
    renderer = column.getHeaderRenderer();

    component =
      renderer.getTableCellRendererComponent
        (__table, null, false, false, 0, 0);

    label = new HeaderRenderer(column);
    label.setText(__COLUMNS[0]);
    label.setIcon(__upIcon);
    label.setHorizontalAlignment(JLabel.LEFT);
    label.setHorizontalTextPosition(JLabel.RIGHT);

    label.setBackground(component.getBackground());
    label.setForeground(component.getForeground());
    label.setFont(component.getFont());

    if(component instanceof JComponent) {
      label.setBorder(((JComponent)component).getBorder());
    }

    column.setHeaderRenderer(label);

    headerListener = new MouseAdapter() {
      private boolean __up = true;

      public void mouseClicked(MouseEvent e) {
int col;
TableColumnModel columnModel = __table.getColumnModel();
        col = columnModel.getColumnIndexAtX(e.getX());
        col = __table.convertColumnIndexToModel(col);
        if(e.getClickCount() == 1 && col == 0) {
          if(__up)
            label.setIcon(__downIcon);
          else
            label.setIcon(__upIcon);
          __up = !__up;
        }
        }
      };


      header = __table.getTableHeader();
      header.addMouseListener(headerListener);
    }


    public static void main(String[] args) {
      Frame frame;
      WindowListener exitListener;

      exitListener = new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
    Window window = e.getWindow();
    window.setVisible(false);
    window.dispose();
    System.exit(0);
        }
      };

      frame = new JTableHeaderIcon();
      frame.addWindowListener(exitListener);
      frame.pack();
      frame.setVisible(true);
    }
  }
```

and fetch the socket send and receive buffer sizes. When setting the buffer sizes, it helps to remember that the value is only a hint to the underlying system TCP/IP code as to what size should be allocated. The underlying system will not necessarily allocate exactly what you request.

**Q** How can a client applet dynamically determine the address of the originating server without hard coding the address in the applet? Also, how can the server application get the address of the connecting applet?

**A** The best way to tell an applet what server it should connect to is to pass the server IP address or hostname and port number as an applet parameter. That way, you only have to change the HTML file if the server changes. If your server is running on the same machine as the Web server from which the applet was downloaded (which is often the case), you can play some tricks with `getCodeBase()` or `getDocumentBase()` to discover the hostname. Either method will return a URL instance from which you can extract the host by calling `getHost()`. However, I don't recommend this approach because it limits your ability to test your applet during development. You have to load the applet from a Web server every time. During development, you will probably want to load it from the local file system using applet viewer or a Web browser. When you do this, the code base and document base URLs will not have a host property. An alternative is to use an applet parameter only for testing purposes, and when you deploy the applet, switch to querying the base URL. But if you do this and you move your server

**This is a common question among C++ programmers moving to Java**

to a host different from your Web server, you will break the applet. You need to choose the solution appropriate to your situation.

To answer your second question, the server application can obtain the address of the connecting applet straight from the socket connection. Your server will be listening for incoming connections using a `ServerSocket`. Accepting a connection with `accept()` produces a `Socket` instance that is used to communicate with the applet. You can get the applet's host information by calling the `getInetAddress` method of that `Socket` instance.

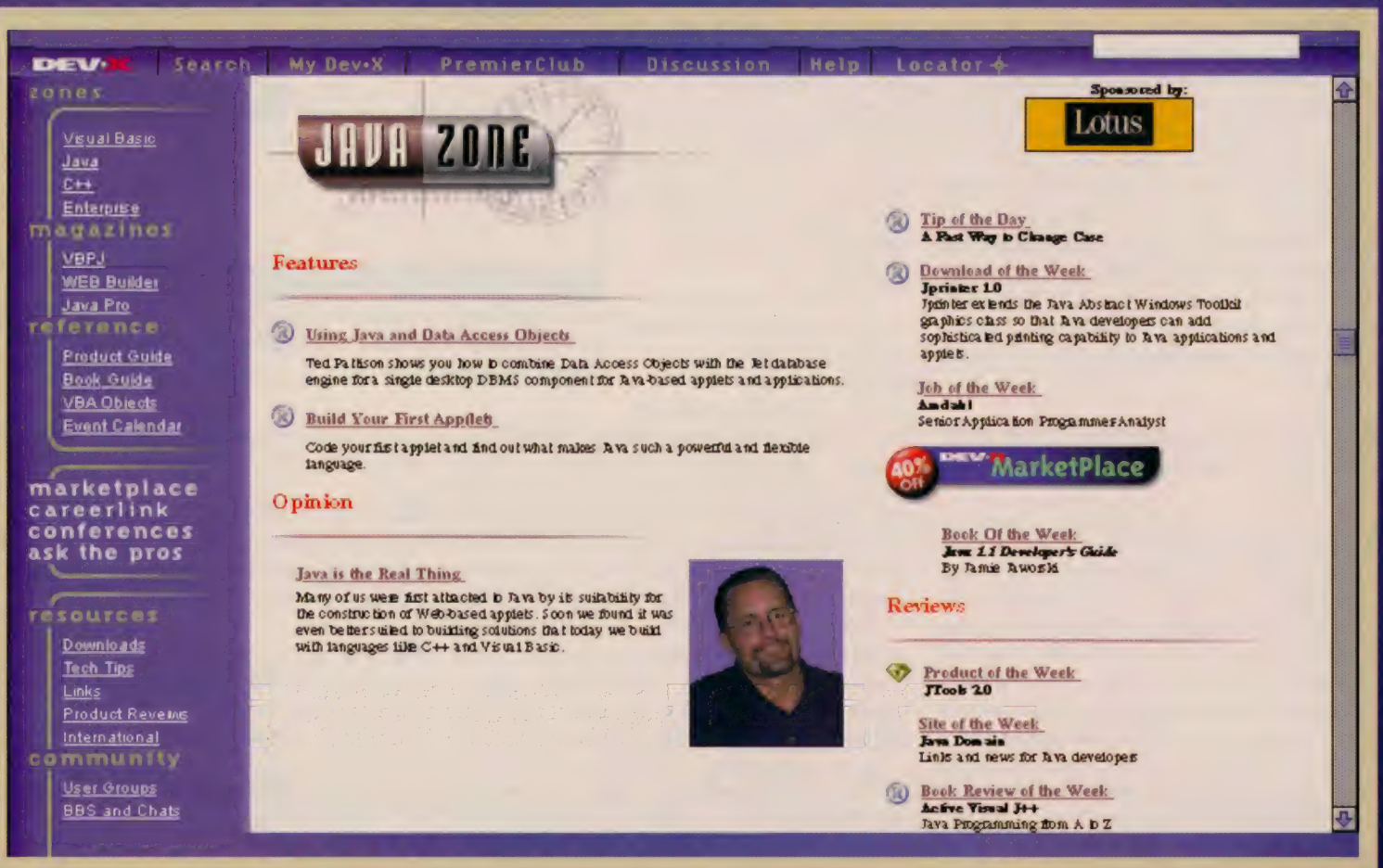
**Q** How do I add an icon to the header of a column of a `JTable` indicating the direction the column is currently sorted?

**A** It can be pretty tough (even for a Java programmer experienced with Swing) to figure out how to change the rendering of column headers. The `TableColumn` class is responsible for storing all the physical attributes of `JTable` columns. The attributes include the renderer for the column header, which is a `TableCellRenderer` instance. Once you know this, all you have to do is get the `JTable`'s `TableColumnModel` with `getTableColumnModel()`, fetch the `TableColumn` for the column whose header you want to alter with `getColumn()`, and set the header renderer for that `TableColumn` using `setHeaderRenderer()` (see Figure 1). Rather than include extraneous code that demonstrates sorting (which can be found in the Swing example code), the code that I provide in Listing 1 only shows you how to change the icon displayed in the label each time the label is clicked by the mouse.

**Q** Is there a `LayoutManager` that would allow me to layout components exactly where I want them? If not, is there another way?

**A** There is no standard AWT or Swing `LayoutManager` implementation that will perform

# www.java-zone.com




## 100% Pure Development Solutions. Delivered Daily Online.

## It's all here on DevX's Java Zone.

Jump on today and get immediate access to the most comprehensive source of expert advice, information, code, and essential resources to help you develop serious Java applications, including:

- **Technical articles and code** from *Java™Pro, Visual Basic Programmer's Journal* and *WEB Builder* magazines
- **Daily programming tips**—plus a library of thousands more
- **Download Java tools**—new applications and applets featured weekly
- **Search Products**—from our database of over 5,000 programming tools
- **Product Reviews**—hundreds of reviews updated weekly
- **Forums**—dedicated to Java development
- **40% off Books of the Week**—plus editors' picks and reviews of top programming titles
- **Conference updates**—register online

**And that's just the beginning.**

DEV·X
The Development Exchange

Explore the rest yourself at
**www.java-zone.com.**

Java-Zone is a member of the family of Windows and interactive development sites located on The Development Exchange at http://www.devx.com.
Java is a registered trademark of Sun Microsystems. The Development Exchange and DevX are trademarks of Fawcette Technical Publications.
Fawcette Technical Publications, 209 Hamilton Avenue, Palo Alto, CA 94301-2500. Phone: 650.833.7100; Fax: 650.853.0230.

absolute positioning of components. Instead, `java.awt.Component`, and therefore every subclass of Component, contains a `setLocation()` method that allows a component to be positioned using coordinates that are interpreted relative to the component's parent container. To absolutely position components without interference from a layout manager, you have to set the layout manager of the container to null. You also have to explicitly set the sizes of your components, not relying on a layout manager to automatically size your components for you. The code in Listing 2 places a button inside a frame at a random location. Every time you press the button, it moves itself to a new random location within the frame (see Figure 2).

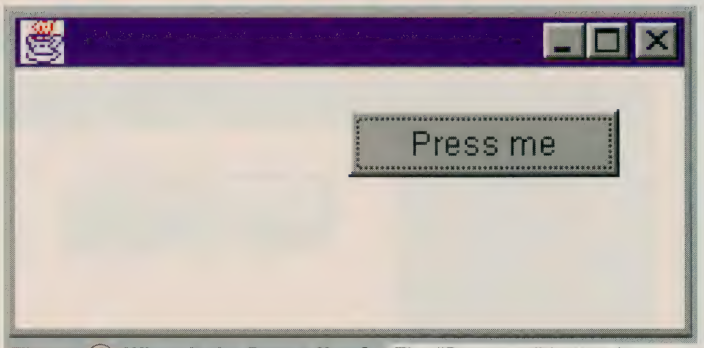



**Figure 2 Where's the Button Now?** The "Press me" button is moved to a randomly selected location each time it is pressed. This is done by setting the container's LayoutManager to null and calling the button's setLocation() method.

**Q** Is there a Java equivalent class or method for Unix Internet address manipulation routines like `inet_ntoa()`?

**A** Java does not have direct analogs for the various `inet` functions that originally appeared in the BSD 4.3 C library, but it supports the equivalent functionality. All Internet address manipu-lation in Java is handled by the `InetAddress` class in the `java.net` package. `InetAddress` instances cannot be created directly using the new operator and invoking a constructor; they can only be created by calling one of the `InetAddress` static methods that perform host lookups: `getLocalHost()`, `getByName()`, and `getAllByName()`. Once you have created an `InetAddress` object, you can obtain the hostname, the string representation of the IP address, or the raw byte representation of the IP address in network byte order, by invoking the `getHostName()`, `getAddress()`, and `getHostAddress()` methods.

All classes and their methods that require IP addresses deal with `InetAddress` objects. Therefore, the `inet_ntoa()` C function you refer to does not quite have a direct analog. Even though calling `getHostAddress()` provides similar functionality, if you only have the byte representation of an IP address, there is no canned way of turning it into an `InetAddress` instance. It is, however, easy to write a method that will do this for you (see Listing 3). The `rawToInetString()` method performs essentially the same task as `inet_ntoa()`, and `rawToInetAddress()` utilizes the result to create an `InetAddress` instance. The main program merely tests the methods by taking a list of hostnames as arguments, looking up their addresses, and using the resulting `InetAddress` objects to verify the `rawToString()` and `rawToInetAddress()` methods.

**Q** Given a reference to an object `B` that is a subclass of `A` and overrides method `foo()`, is there anyway I can invoke the base implementation of `foo()` directly?

**A** This is a common question among C++ programmers moving to Java. C++ provides a special syntax for allowing subclasses to invoke overridden methods declared anywhere above them in the class hierarchy. Java does not provide a similar mechanism, with the exception of constructors (which are not regular

**Listing 2 Absolute Positioning** By setting a container's LayoutManager to null, you can explicitly position its child components using setLocation().

```
import java.awt.*;
import java.awt.event.*;

public final class AbsoluteLayout {

  public static int random(int min, int range) {
    double dRange = range;
    double result;

    result = Math.rint(Math.random()*dRange);
    return (int)(min + (int)result);
  }

  public static void main(String[] args) {
    Button button;
    Frame frame;
    WindowListener exitListener;

    exitListener = new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
  Window window = e.getWindow();
  window.setVisible(false);
  window.dispose();
  System.exit(0);
      }
    };

    button = new Button("Press me");

    // Explicitly size the button, otherwise it will be a
    // mere point.
    button.setSize(100, 25);

    // Add an ActionListener that will randomly move the
    // button
    button.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
  Component source, parent;
  Dimension sourceSize, parentSize;

  source = (Component)e.getSource();
  sourceSize = source.getSize();
  parent = source.getParent();
  parentSize = parent.getSize();
  source.setLocation(random(0, parentSize.width -
    sourceSize.width),
        random(0, parentSize.height - sourceSize.height));
      }
    });


    frame = new Frame();

    // Don't forget to set the layout manager to null!
    // Otherwise it will interfere with your positioning and
    // sizing.
    frame.setLayout(null);
    frame.add(button);
    frame.addWindowListener(exitListener);
    frame.setSize(400, 400);

    // Dispatch an initial event to randomly place
    // the button.
    button.dispatchEvent(
       new ActionEvent(button,
           ActionEvent.ACTION_PERFORMED, null));
    frame.setVisible(true);
  }
}
```

**Listing 3 Converting Addresses** With a little bit of work you can convert the byte representation of an IP address to a String, much like C's inet_ntoa().

```
import java.net.*;

public final class RawIPToString {

  // Mimics BSD Unix's inet_ntoa(), assumes address is in
  // network byte order and that the array has 4 entries.
  public static String rawToString(byte[] address) {
    int octet, count = 0;
    StringBuffer result = new StringBuffer();

    while(true) {
      octet = address[count] & 0xff;
      result.append(octet);
      if(count++ >= 3)
  break;
      result.append('.');
    }

    return result.toString();
  }

  public static InetAddress rawToInetAddress(byte[] address)
    throws UnknownHostException
  {
    return InetAddress.getByName(rawToString(address));
  }

  public static void main(String[] args) {
    int host;
    InetAddress address, fabricatedAddress;

    if(args.length < 1) {
      System.err.println("Usage: RawIPToString hostname1
        [hostname2] ...");
      System.exit(1);
    }

for(host = 0; host < args.length; ++host) {
      try {
  address = InetAddress.getByName(args[host]);
      } catch(UnknownHostException e) {
  System.err.println("Unknown host: " + args[host]);
  continue;
      }
      System.out.println("getAddress() : " + args[host]
        + " is " +
       address.getHostAddress());
      System.out.println("rawToString(): " + args[host]
        + " is " +
       rawToString(address.getAddress()));

      try {
  fabricatedAddress =
    rawToInetAddress(address.getAddress());
      } catch(UnknownHostException e) {
  System.err.println(
       "Could not fabricate an address with
        rawToInetAddress()");
  continue;
      }

      if(fabricatedAddress.equals(address))
  System.out.println("Fabricated and original addresses
    are equal.");
    else
  System.out.println(
       "Error: Fabricated and original addresses are
        NOT equal.");
    }
  }
}
```

**Listing 4 Method Inheritance** Unlike C++, you can't access a method in a superclass that is overriden by a subclass. All Java methods are virtual.

```
class A {
  public int value;

  public String foo() { return "A: foo"; }
}

class B extends A {
  // This lexically hides the member variable in A
  public int value;
  // This overrides the method in class A
  public String foo() { return "B: foo"; }
}

public final class Override {

  public static final void main(String[] args) {
    B b = new B();
    A a = b;

    a.value = 2;
    b.value = 3;

    // Lexically hidden member variables can be accessed
    // through upcasting
    System.out.println("A value: " + a.value);
    System.out.println("B value: " + b.value);
    System.out.println("A value through upcasting: "
      + ((A)b).value);

    // All three of these will print "B: foo" because all
    // Java methods are virtual and once overridden cannot
    // be recovered.
    System.out.println("\nA foo(): " + a.foo());
    System.out.println("B foo(): " + b.foo());
    System.out.println("A foo through upcasting: "
      + ((A)b).foo());

  }
}
```

**Listing 5 Serving Clients** It is usually better to aggregate your communication when writing client/server applications such as this RMI list update example.

```
/***
 * $RCSfile: ListData.java,v $
 ***/

import java.rmi.*;

public interface ListData extends Remote {
  public String[] getData() throws RemoteException;
}


/***
 * $RCSfile: ListDataImpl.java,v $
 ***/

import java.net.*;
import java.rmi.*;
import java.rmi.server.*;

public class ListDataImpl extends UnicastRemoteObject
  implements ListData {
  String[] _data = { "1", "2", "3", "4", "5" };

  public ListDataImpl() throws RemoteException { }

  public String[] getData() throws RemoteException {
    return _data;
  }

  public static final void main(String args[]) {
```

*Continued on next page.*

*Continued from previous page.*

```
    String hostname;
    ListDataImpl listData;

    try {
      InetAddress address = InetAddress.getLocalHost();
      hostname = address.getHostName();
    } catch(java.net.UnknownHostException e) {
      e.printStackTrace();
      return;
    }

    System.setSecurityManager(new RMISecurityManager());

    try {
      String url = "//" + hostname + "/ListServer";
      listData = new ListDataImpl();
      Naming.rebind(url, listData);
      System.out.println("Bound ListServer in registry: "
        + url);
    } catch(Exception e) {
      e.printStackTrace();
      return;
    }
  }
}


/***
 * $RCSfile: ClientList.java,v $
 ***/

import java.awt.*;
import java.awt.event.*;

public class ClientList extends List {

  public ClientList() { }

  public ClientList(int rows) { super(rows); }

  public ClientList(int rows, boolean multipleMode) {
    super(rows, multipleMode);
  }

  public synchronized void setData(String[] data) {
    removeAll();
    for(int i=0; i < data.length; i++)
      add(data[i]);
  }

}


/***
 * $RCSfile: UpdateList.java,v $
 ***/

import java.awt.*;
import java.awt.event.*;

import java.rmi.*;

public final class UpdateList {
  public static final String[] DATA = {
    "One", "Two", "Three", "Four", "Five"
  };

  public static void main(String[] args) {
    ClientList list;
    Frame frame;
    WindowListener exitListener;
    String hostname;
    ListData listData;

    if(args.length < 1) {
      System.err.println("Usage: UpdateList
        <RMI Server Hostname>");
      return;
    }

    hostname = args[0];

    exitListener = new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
  Window window = e.getWindow();
  window.setVisible(false);
  window.dispose();
  System.exit(0);
      }
    };

    list = new ClientList(4);
    frame = new Frame();
    frame.addWindowListener(exitListener);
    frame.add(list);
    frame.pack();
    frame.setVisible(true);

    list.setData(DATA);

    // Get remote data and update list.
    try {
      listData = (ListData)Naming.lookup("//" + hostname
        + "/ListServer");
      list.setData(listData.getData());
    } catch(Exception e) {
      e.printStackTrace();
      return;
    }
  }
}
```

methods), that can explicitly invoke the constructor of the parent class using the special `super()` invocation syntax. However, Java does allow lexically hidden member variables to be accessed through upcasting. The same technique cannot be used as a workaround to invoke overridden methods belonging to a parent class because all Java methods are virtual. If you upcast an object and call an overridden method, the result will be for the method defined as far down as possible in the class tree to be called. The example program in Listing 4 uses upcasting to access a member variable, demonstrating why the same technique will fail for invoking methods.

## Code Online

All the code in this article is available for free at www.java-pro.com. Use the Locator+ code shown below to download the code. Free online registration is required. See the Online Resources box in "Letters to the Editor" for more information.

**Are We There Yet?**
Registered Locator+: **JP0299**
Premier Locator+: **JP0299AP**

**Q** What's the best way for my RMI server application to populate a `java.awt.List` with data on the client with a method invocation?

**A** When writing distributed applications, it is usually desirable to minimize the amount of communication. This means you want to reduce the number of messages and aggregate data into one message when possible. It is generally better to send a few large messages than a lot of little ones. It can often be difficult to do this when using distributed objects because the programming model tends to make code look just like a non-distributed program. In your situation, you want to avoid having the server modify the client's list one item at a time. Rather, you want to send all the data to the client, and let the client update the list.

In Listing 5, `ListData` and `ListDataImpl` comprise the server. After compiling these, rmic must be run on `ListDataImpl`. The main client program is contained in `UpdateList`, which uses the `ClientList` class. ■

JAVA Pro

# Readers Poll

## YOUR VIEWS ON *JAVA PRO*:

Your feedback is vital to us. Taking a minute to fill out this questionnaire will help us to refine our direction and content. This survey is also available online at www.java-pro.com. Tell us how you use Java and what you want from *Java Pro*. Thank you.

**1. How do you rate this issue?**

| | | | | | |
|---|---|---|---|---|---|
| Overall rating | ❑ Excellent | ❑ Good | ❑ Fair | ❑ Poor | |
| Technical content | ❑ Too Technical | ❑ Just Right | ❑ Not Deep Enough | | |
| Usefulness in your work | ❑ 5 (Very Useful) | ❑ 4 | ❑ 3 | ❑ 2 | ❑ 1 (Not Useful) |

| | Excellent | Good | Fair | Poor |
|---|---|---|---|---|
| **2. How do you rate each editorial feature?** | | | | |
| A Test Panel for a Pluggable Look and Feel | ❑ | ❑ | ❑ | ❑ |
| A SortFactory for the Collections API | ❑ | ❑ | ❑ | ❑ |
| Introduction to the Datagram Protocol | ❑ | ❑ | ❑ | ❑ |
| Dynamic Method Invocation with CORBA and Java | ❑ | ❑ | ❑ | ❑ |
| **3. How do you rate each column?** | | | | |
| Ask the Java Pro | ❑ | ❑ | ❑ | ❑ |
| Javatecture | ❑ | ❑ | ❑ | ❑ |
| Impure Thoughts | ❑ | ❑ | ❑ | ❑ |
| Servlet Solutions | ❑ | ❑ | ❑ | ❑ |
| VM Roadtest | ❑ | ❑ | ❑ | ❑ |

## HOW DO YOU USE JAVA?

**4. Do you develop JavaBeans for use by others?**

❑ Yes, for internal use ❑ Yes, for commercial use ❑ No

**5. How would you describe your Java usage?**

❑ Just evaluating Java ❑ Small-scale development ❑ Large-scale development ❑ Not Using

**6. What types of Java applications do you develop?**

❑ Browser-based Internet ❑ Browser-based intranet ❑ Stand-alone executable
❑ Server applications ❑ *n*-tier Client/Server

**7. What areas are you most interested in?**

❑ Database access ❑ CORBA ❑ XML/HTML ❑ Servlets ❑ Stand-alone apps

**8. Is developing "100% Pure" Java applications important to you and/or your organization?**

❑ 100% Important ❑ 75% Important ❑ 50% Important ❑ Gimme J/Direct

Thank you for your help. Please photocopy this page and mail it or fax it to: **650-853-0230**

**FAWCETTE TECHNICAL PUBLICATIONS • 209 HAMILTON AVE • PALO ALTO, CALIFORNIA 94301-2500 • USA**

Javatecture

# Using Exceptions Effectively

## Keep your apps running even when they're processing dirty data

by James W. Cooper

Recently I talked with Peter Haggar and read some comments by Ken Arnold about exceptions and it got me thinking about how little we do with exceptions in Java programs. I've seen few examples of how to use exceptions effectively in actual programs and the advice in most programming texts and articles is brief.

To review, the Java runtime system throws exceptions for a number of error conditions, such as *no such file*, *read past end of file*, *input errors*, *number format error*, *array index out of bounds*, and *null pointer*. There are, of course, dozens of other exceptions, especially in the Java Foundation Classes. Most input and output class methods throw exceptions, and, if the method declares that it throws an exception, the Java compiler requires you to catch that exception. This is the general form of such exceptions:

```
try{
  //statements...
}
catch (SomeException e)
{
  //handle exception
}
finally
{
  //code always executed
}
```

If one of the statements inside the `try` block fails and throws an exception, the code inside the `catch` block is executed. Then, whether there is a failure or not, code inside the `finally` block is executed.

You can also catch a number of different exceptions and deal with them in separate `catch` blocks:

```
try {
 //open some sort of file
}
catch(FileNotFoundException fn)
  { //handle file not found
}
catch (EOFException e)
  { //handle eof
```

| | Swimmer | Age | Club | Time |
|---|---|---|---|---|
| 1 | Amanda Mc Carthy | 12 | WCA | 29.28 |
| 2 | Jamie Falco | 12 | HNHS | 29.80 |
| 3 | Meaghan O' Donnell | 12 | EDST | 30.00 |
| 4 | Greer Gibbs | I2 | CDEV | 30.04 |
| 5 | Rhiannon Jeffrey | 11 | WYW | 30.04 |
| 6 | Sophie Connolly | 12 | WAC | |
| | | | | 30.05 |
| 7 | Dana Helyer | 12 | ARAC | 30.18 |
| 8 | Lindsay Marotto | 12 | OAK | 30.23 |
| 9 | Sarah Treichel | 12 | WYW | 30.35 |
| 10 | Ashley McEntee | 12 | RAC | |

**Table 1 Expecting the Exceptions** A close look at the data in this table reveals a number of minor typographical errors that would cause a computer program to go bananas. By handling exceptions effectively in our program, we can avoid the problems these errors could cause.

James W. Cooper is a computer science researcher and the author of a number of books in the field. He is currently working on his 13th book, on Java Design Patterns, for Addison Wesley Longman.

```
  }
catch (IOException e)
{ //handle general IO failure
}
```

It is important, however, that you start with the most-specific, derived exception and end with the most-general exception.

Too often, we write exception-handling code to be too simple—if the compiler compels us to catch an exception, we do the minimum we can get away with:

```
try {
f = new RandomAccessFile
      (fname, "r");
}
catch (IOException e)
{System.out.println("Can't open file");
}
```

And usually, printing out a message is the least useful thing you can do, especially in a GUI windowing system, where the printed message might not show up where anyone will ever see it. Worse yet, the effect of some of these exceptions is to make the program unusable.

With these issues in mind, I decided to write a program that uses a couple of exceptions in the manner that the designers of Java intended, and keeps running even though the input data might contain some sloppy errors.

## An Input Data File

Let's assume that we have a set of data representing results of a swim meet that we need to read into a program. We'll further suppose that this data was typed into a file manually and therefore contains some typing errors. The data in Table 1 is perfectly readable for human beings, but contains, the kind of errors that cause a computer program to throw multiple exceptions. For example, although we assume each swimmer will have just one first and one last name, two swimmers have spaces in the middle of their last names. The fourth swimmer has a lowercase "L" instead of a one for the first digit of her age. The fifth swimmer has uppercase "Os" instead of zeroes in her time. The sixth swimmer has an extra line feed between the club initials and her time, and the last swimmer's time is missing. Most of these problems will generate exceptions if we try to read in each line and assign the data to a particular swimmer. We could catch those exceptions and throw up our hands in defeat, or we could try to be a little more thoughtful in figuring out how to recover from these errors.

**Listing 1 Handling Error-Filled Data** Data with typos can be managed by creating a Kid class that parses each line of the file and creates a Kid object, which contains as many of the data values as it can figure out.

```
public Kid(String s)   {
  tok = new saveTokenizer(s);
  //must be at least one
  if(tok.hasMoreElements()){
     legal = true;       //Start as legal
    //discard line number
     String lnum = tok.nextToken();
     getFrname();        //read first name
     getLname();         //last name
     getAge();           //convert age
     getClub();          //read club
     getTime();          //convert time
   }
     else
        legal = false;  //otherwise not legal kid
  }
//---------------
private void getFrname()   {
  if(tok.hasMoreElements())
      fr_name = tok.nextToken();
  else
     legal = false;
}
//---------------
  private void getLname()   {
    if(tok.hasMoreElements())
    l_name = tok.nextToken();
  else
    legal = false;
}
```

Exceptions in Java are intended to handle unexpected occurrences. If you can test for a condition (such as end of file) directly in your code, then an exception is the wrong way to handle the condition. You should realize that each exception thrown can reduce the performance of the program. This suggests that *creating* a lot of new exceptions within your class hierarchy could be a bad idea. On the other hand, *using* exceptions that the system provides or requires is good coding practice, and, in many instances, is enforced by the compiler.

We should note, to be completely accurate, that the Java compiler requires you to catch all exceptions *except* those derived from the RuntimeException class. The run-time exceptions might occur anywhere, and they include division by zero and the like.

## A Kid Class

In order to handle this "dirty data," we'll create a `Kid` class that parses each line of the file and creates a `Kid` object that contains as many of the data values as it can figure out. The way to handle our data ingestion problem is to create a `Kid` object for each line in the file and check each line in a separate method. If it can't find at least a first name, last name, and club, it sets the `legal` state to `false` and skips over that line of data.

We need to look ahead to the next token without committing ourselves to its use so that we can see if the last name contains a space. So we derive the `saveTokenizer` class from `StringTokenizer`. This class contains a `pushToken()` method that puts a token back if it isn't the one we need.

Our basic `Kid` class constructor takes a line from the data file and parses it into the names, age, club, and time values

## Resources


Cooper, James; *Principles of Object-Oriented Programming in Java 1.1*, Scottsdale, AZ: Ventana/Coriolis, 1997.

Gosling, James; Yellin, Frank; *The Java Team, The Java Application Programmng Interface, Vol. 1, Core Packages*, Reading, MA: Addison-Wesley, 1996.

Haggar, Peter; "Java Exception Handling" IBM e-Business and Networking Systems Technical Conference, Las Vegas, September 1998.

**Listing ② Catching Errors** Using an exception mechanism, the catch portion of the code allows us to recover from a number-conversion error.

```
private void getAge()   {
     String agstring= "";
     try{
        agstring = tok.nextToken();
        kid_age = new Integer(agstring).intValue();
     }
     catch(NumberFormatException e)      {
        kid_age = 0;    //no legal age
        //look ahead for next token as legal
        testInteger testAge =
          new testInteger(tok.nextToken());
        if (testAge.isLegal())           {
            //age is in next token
           kid_age = testAge.toInteger();
           l_name += agstring;  //this is part of last name
           }
        else     {
           kid_age = 0;       //no age
           tok.pushToken();   //push next token back
        }
     }
     catch(NoSuchElementException n)
     {legal = false;}
    }
```

(see Listing 1). There is nothing special about reading the first and last name—either they are there or they aren't.

When we read the age token, it is possible that the first token that we encounter is really part of a last name that contains an intervening space. If this token is not numeric, we add it to the last name and get the next token for the age. Listing 2 provides an excellent example of using the exception mechanism to catch a number conversion error and how we can recover from it in the `catch` portion of the code. If the token is not a number, the exception is triggered. If the following token is a number, then the age is set to that number and the previous token appended to the last name. If the following token is not an integer then it is pushed back to be reread. This would occur if the actual age were mistyped and not convertible to an integer.

This routine illustrates two ways of dealing with integer conversion errors. We also create an instance of the `testInteger` class, which encapsulates the `Integer` class and allows you to ask if the token it has just engulfed is an integer or not. Note that because the `Integer` class is a `final` class we can't derive `testInteger` from it but must encapsulate `Integer` and bring some of its methods to the surface of the enclosing class (see Listing 3, available online).

Exceptions

| Fname | Lname | Age | Club | Time |
|---|---|---|---|---|
| Amanda | McCarthy | 12 | WCA | 29.28 |
| Jamie | Falco | 12 | HNHS | 29.8 |
| Meaghan | O'Donnell | 12 | EDST | 30.0 |
| Greer | Gibbs | 0 | CDEV | 30.04 |
| Rhiannon | Jeffrey | 11 | WYW | 0.0 |
| Sophie | Connolly | 12 | WAC | 0.0 |
| Dana | Helyer | 12 | ARAC | 30.18 |
| Lindsay | Marotto | 12 | OAK | 30.23 |
| Sarah | Treichel | 12 | WYW | 30.35 |
| Ashley | McEntee | 12 | RAC | 0.0 |

**Figure ① Organizing the Kids** By handling exceptions effectively, we can take data with mistakes and load it into a data table and display it correctly.

## Keeping with the Times

There is nothing special about how we get the club token. If there aren't any tokens left, the legal flag is set to false. The time method works much like the age conversion; we catch the float conversion exception and set the resulting time to zero. We don't make the entire swimmer's data illegal however, because we can keep the swimmer and correct the time value later:

```
private void getTime()   {
     Float tim;
try{
      tim = new Float(tok.nextToken());
    kid_time = tim.floatValue();
     }
catch (NumberFormatException e)
   {kid_time = 0;}
     catch(NoSuchElementException e)
     {kid_time = 0;}
  }
```

Note that if we run out of tokens in the tokenizer, we catch the NoSuchElement exception and a zero time is recorded.

## Showing the Results

Our main program loads all of these `Kid` objects into a Table Data model and displays them as shown in Figure 1:

```
KidModel tmodel = new KidModel();
//table data model
InputFile f = new InputFile("50free.txt");
String s = f.readLine();
while( s != null)
  {
  k = new Kid(s);
  if(k.isLegal())
     tmodel.add(k);
  s = f.readLine();
  }
JTable table = new JTable(tmodel);
```

You can see that we have successfully dealt with the typos in our data. The two last names with spaces have been rejoined, however, the 4th age is indicated as zero and the 5th, 6th, and 10th times as zero. This means that they probably require manual intervention to correct.

You might have noticed that we introduced the InputFile class. This is itself a simple class that encloses file I/O exceptions and we have discussed it previously (see "References").

Exceptions don't always lead to rapid program exits. If they did, we could just leave them uncaught. Instead we've seen here how to use exception handling to improve error checking and the robustness of your code in the face of faulty user input. ■

## Code Online

All the code in this article is available for free at www.java-pro.com. Use the Locator+ codes shown below to download the code. Free online registration is required. See the Online Resources box in "Letters to the Editor" for more information.

**Using Exceptions Effectively**
Registered Locator+: **JP0299**
Premier Locator+: **JP0299JT**

# *Finally, a Magazine for Programmers Building Serious Java Business Applications*

# AND NOW JAVA™ Pro IS MONTHLY!

***Java Pro* magazine** is the first independent technical magazine dedicated to serious Java development.

*Java Pro* is the professional developer's independent source for information on building serious Java applications. *Java Pro* takes you beyond the hype to deliver hands-on, code-intensive, product-oriented information to help you harness Java's unique capabilities in your enterprise-wide development efforts. Each month, *Java Pro* tackles tough development issues and provides unbiased, no-nonsense coverage of the latest tools on the market.

**Here's what you can expect from *Java Pro*:**

- Practical techniques for enterprise development
- Tips and tricks for fine-tuning Java performance
- Independent product reviews and new-product announcements
- Complete source-code examples using JavaBeans, servlets, and more!

**Don't miss this special offer: Save 58% off the cover price!**

Call today to subscribe to *Java Pro*. You'll pay the low rate of $29.95 for a full year–twelve issues in all. That's a savings of more than 58% off the newsstand rate!

*Java Pro* comes to you from Fawcette Technical Publications, the creators of *Visual Basic Programmer's Journal* and *WEB Builder*. We provide technical information to more developers than anyone in the world.

**Money-back guarantee**

If you are not satisfied with ***Java Pro*** at any time, ***we will refund the cost of your subscription.*** No questions asked.

**Call Today:**

**1.800.848.5523**

Outside the U.S., call: **650.833.7100**

Order online: **http://www.java-pro.com**

FTP FAWCETTE TECHNICAL PUBLICATIONS
209 Hamilton Avenue
Palo Alto, CA 94301-2500

One year is twelve issues. Annual cover price is $71.40. Subscription begins upon receipt of payment. Please allow 4-6 weeks for delivery of first issue. International subscriptions must be payable in U.S. dollars plus postage. Canada/Mexico add $18/year; all other countries add $44/year for airmail. Canadian GST included.

Java is a trademark of Sun Microsystems, Inc.

# Inverted Priorities

## A dark facet of the multithreaded gem—priority inversion makes a low-priority thread run as though it has enjoyed a priority boost

by Rick Grehan

Rick Grehan is a programmer/writer in the Discover Products group at Metrowerks Inc. Rick was a Senior Editor at *BYTE* magazine (where he wrote the "JavaTalk" column) and at *Computer Design* magazine. He can be reached at rgrehan@top.monad.net.

On last month's column, I opened with a discussion of the various multithreading architectures. This month, I've chosen to explore one of the darker aspects of multithreaded programming—priority inversion. (I wrote this column around Halloween. Perhaps there's a connection...)

There are a couple of things to keep in mind. I'll be speaking of "ownership" of resources. This means acquiring an exclusive lock on a system resource such as a shared variable, a file, or a critical section of code, such that only one thread has access to the locked item. The locked item only becomes available to other threads when the item is unlocked. Programming languages and operating systems provide a variety of means for implementing such locks. Java provides synchronized objects, methods, or blocks of code.

I will presume that readers already have a good grasp of priorities and scheduling. Here's a quick review: Java threads have ten priority levels. Within each priority level, threads are scheduled to run in round-robin fashion. When it's time for the scheduler to pick the next thread to run (because of a time slice conclusion or the current thread becoming blocked) the scheduler will run the highest-level priority task that's ready.

### Deadlock and Priority Inversion

It is important that you don't confuse deadlock with priority inversion. Deadlock occurs when task A owns resource 1 and wants resource 2. Meanwhile, task B owns resource 2 and wants resource 1. The two tasks will not release their respective resource and will lock up.

Avoiding deadlock requires preventive programming. Don't hope that if deadlock materializes in your program you can locate it and program your way out of it. Deadlock is often hard to recognize—program so it doesn't happen in the first place.

Some anti-deadlock programming habits are obvious: every `wait()` should have a corresponding `notify()`. (Neatness counts even in programming. If you take something out, put it back.) Some are less obvious: Don't make any blocking calls from within a synchronized object, or a "thread pileup" can result. Serialize access to synchronized objects so that all threads that access those objects do so in the same order. If thread A accesses synchronized object 1 and then object 2; that ordering should be used by all threads that access those objects. Similarly, threads should release objects in LIFO order—object 2 released, then object 1. Incorporating these habits into your programming will save you a lot of headaches.

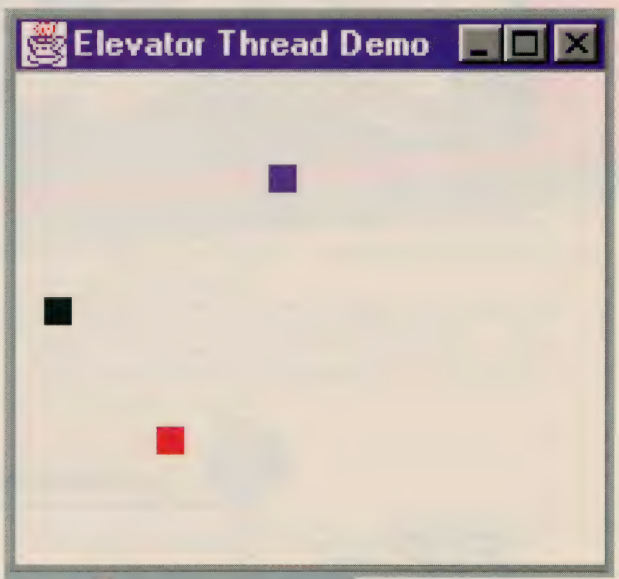



**Figure 1 Stuck Between Floors** Each square represents an elevator process. The first and last elevators share a process. Once the middle process starts, priority inversion locks up the first and last process.

Priority inversion is a peculiar malaise that can occur in multitasking systems that support task priorities (which is almost any multitasking system). It is particularly deadly in real-time multitasking systems, where a task of a given priority *must* run within strict time intervals or the whole system collapses. (The Mars Lander software suffered from priority-inversion problems, until the engineers diagnosed and corrected it. You can find a good account of the debugging process at www.java.sun.com/people/jag/pathfinder.html). As you'll see, priority inversion can also happen in desktop systems... yours, for example.

Priority inversion is an unusual interaction among threads of differing priorities. The priorities of a pair of threads get "inverted," and a low-priority thread begins to act like a high-priority thread. Meanwhile, the high-priority thread is, through no fault of its own, stuck waiting for a low-priority thread. The low-priority thread is stuck too, because the computer can't figure out what's happened. That's where you come in.

## Under the Hood

The best way to illustrate priority inversion is with an example. Suppose you have three active threads, each running at a different priority level. Thread A has highest priority, thread B is in the middle, and thread C runs at lowest priority. (The priority levels are relative to one another.) Furthermore, suppose you've defined a synchronized object that guards ownership of a shared resource—such as a shared variable—in such a way that only one thread at a time can own that resource. Threads A and C vie for ownership of that resource.

**Deadlock is often hard to recognize—program so it doesn't happen in the first place**

At some point, thread C is executing, acquires the resource, and, while still in ownership of the resource, is swapped out by the scheduler, because thread A is ready to run. Thread A is scheduled, but it is almost immediately blocked because it attempts to acquire the guarded resource (now owned by C).

Thread B starts running, and priority inversion sets in. As long as thread B is unblocked, thread A cannot run because thread C still owns the resource. And, thread C cannot run because higher-

**Listing 1 The Elevator Thread Class** Each elevator runs at a different priority. The high- and low-priority elevators contend for the binary semaphore. Note that, rather than use the thread priority to select calls to the semaphore, I simply use the elevator's onscreen "slot." Only the high-priority elevator sleeps, which gives the low-priority elevator time to run before the middle elevator is started.

```
import java.awt.*;

class Elevator extends Thread
{
  // Constructor for Elevator.
  // The argument "place" selects which "position" the
  // elevator is created in (horizontal position, that is).
  // We'll also use it to set the priority of the thread
  // driving elevator, such that the leftmost elevator has
  // the highest priority, and the rightmost elevator has
  // the lowest.
  public Elevator(Canvas c, int place, int priority,
    Color col, BinarySemaphore bsema)
  {
    box=c;  // Get reference to Canvas
    x=place; // Set location onscreen
    p=priority;  // Set thread's priority
    clr=col; // Set box color
    bs=bsema;  // Set binary semaphore reference
  }

  public void run()
  {
    double k=.5;

    // Set our priority
    try {
      setPriority(p);
    }
    catch (IllegalArgumentException ex)
    { }

    // Draw the first instance of the elevator.
    Graphics g=box.getGraphics();
    g.setColor(clr);
    g.fillRect(x,y,XSIZE, YSIZE);
    g.dispose();

    // We won't run forever. Just for a few thousand
    // iterations.
    for(int i=1; i<=3000; i++)
    {
      move();

      // Only the high-priority thread sleeps. The other
      // threads waste time in calculations. In addition, the
      // high and low priority threads lock and unlock the
      // Semaphore
      if(x==10)
      {
        try {
          // This setting for sleep() gives the low-priority
          // elevator enough CPU time to run as well...until
          // the medium-priority elevator starts up. Making
          // this number smaller gives the high-priority
          // thread more runtime, and it will begin to hog
          // the CPU.
          Thread.sleep(10);
        }
          catch (InterruptedException ex)
          { }

      }

      // High and low priority lock the resource
      if((x==10) || (x==90))
        bs.lock();
      // All threads eat time calculating
       for(int j=1; j<=500000; j++)
        for(int l=1; l<=5; l++)
          k=Math.sin(k);
      // High and low priority unlock the resource
      if((x==10) || (x==90))
        bs.unlock();

    }
  }
```

Continued on next page.

Continued from previous page.

```
  public void move()
  {
    Graphics g=box.getGraphics();
    Dimension d=box.getSize();

    g.setColor(clr);
    g.clearRect(x,y,XSIZE,YSIZE); // Erase the old

    y+=yincr;           // Advance elevator

    // If the elevator has crashed into the top,
    // set its location to the top and reverse
    // direction.
    if (y<0)
    { y=0;
      yincr=-yincr;
    }

    // If the elevator has crashed into the bottom,
    // set its location to the bottom and reverse
    // its direction.
    if (y+YSIZE>=d.height)
    { y=d.height-YSIZE;
      yincr=-yincr;
    }

    // Draw the new rectangle and exit
    g.fillRect(x,y,XSIZE,YSIZE);
    g.dispose();    // Release Graphics object
  }

  private Canvas box;
  private static final int XSIZE=10;  // Box size
  private static final int YSIZE=10;
  private int x=0;          // Box location
  private int y=0;
  private int yincr=2;        // Move increment
  private int p;            // Thread priority
  private Color clr;          // Box color
  private BinarySemaphore bs;   // Binary semaphore
object
}
```

priority thread B is unblocked, so the scheduler gives it all the CPU's attention.

### Sickness in Action

I've created a simple application to illustrate priority inversion at work. This application is an onscreen simulation of three elevators (see Figure 1). Each elevator is represented by a colored rectangle, created with the `Elevator` class (see Listing 1) and controlled by a separate thread. The left elevator (black) runs at highest (relative) priority, the middle (red) at middle priority, and the right (blue), at lowest.

The application also contains a class called `BinarySemaphore`, which implements a simple binary semaphore:

```
class BinarySemaphore
{
  public void BinarySemaphore()
  {
    isLocked=false;
  }

  public synchronized void lock()
  {
    while(isLocked)
    { try {
        this.wait();
      }
      catch (InterruptedException
        ex) { }
    }
    isLocked=true;
  }

  public synchronized void unlock()
  {
    isLocked=false;
    this.notify();
  }

  private boolean isLocked;
```

This binary semaphore is a synchronization object that allows only one task to own the semaphore at a time. Binary semaphores can be used as fundamental building blocks for synchronization objects, such as mutexes, critical sections, and counting semaphores. Here it acts as the "bone of contention" for the high- and low-priority threads. Only one thread can "own" the semaphore at a time; any other thread attempting to acquire ownership is blocked.

**When I first built the application, I foolishly set the high-priority thread to PRIORITY_MAX. Don't do that**

The primary ingredients for priority inversion are a high-priority task and a low-priority task contending for exclusive access to a resource, with a medium-priority task (which does not access the resource) thrown in. Consequently, I've written the program so that the high- and low-priority threads grab the semaphore for a brief period of time after moving the elevator. The medium-priority thread does not access the semaphore, and therefore runs unimpeded.

Run the code in Listing 2, and you'll see priority inversion in action. (I ran the code on a Windows NT system using Metrowerks' CodeWarrior development environment, which at the time of this writing uses the JDK 1.1.6 as its runtime.) I've written the code so that the high- and low-priority elevator threads are started first and allowed to run for a few seconds before the medium-priority elevator thread is started.

On my system, priority inversion occurred as soon as the middle elevator came on stage. The high- and low-priority elevators came to a virtual standstill, while the medium-priority elevator continued to run. Occasionally, the low-priority elevator was able to advance a pixel or so, and, even more rarely, the high-priority elevator was able to advance several pixels before freezing again.

A word of caution: I chose the priorities for the threads based on the way the application performed on Windows NT. When I first built the application, I foolishly ignored my own advice from last month's column and set the high-priority thread to `PRIORITY_MAX`. Don't do that.

Under NT, Java threads are mapped to kernel threads. Setting a thread to `PRIORITY_MAX` places the thread at a truly high priority, compared even to threads outside of the Java environment. The result was that the rest of the system came to a near standstill as the high-priority thread consumed nearly all of the CPU bandwidth. Once I killed the process, which took a while, everything returned to normal.

I ignored more of my own advice in regards to programming for portability. Specifically, last month I mentioned that you can't depend on a JVM environment to have a preemptive scheduler, and you must therefore build threads that relinquish the CPU at a regular interval. Otherwise, in a non-preemptive JVM such CPU-hogging threads will starve out other threads.

In fact, the thread controlling the middle elevator hogs the CPU. It doesn't block, `sleep()`, or `yield()`. In my own defense, I had to do that for the example to work. Had I placed a `sleep()` call in that thread, it would have given the low-priority thread a chance to run, which would in turn free up the high-priority thread. (Insert a call to sleep within an `if(x==50)` clause and see what happens.)

Therein lies the cure for the priority inversion problem, and you knew it all along. As you write multithreaded applications, keep in mind that this application might be executed on a non-preemptive JVM, and a few strategically-located `sleep()`, `yield()`, or blocking I/O calls will make all the difference. (However, be careful using `yield()`; it will only allow threads of equal or higher priority to run. `Sleep()` blocks the current thread and permits threads at all priorities to run.)

## It's Up to Us

I stated earlier that priority inversion can occur in almost any multitasking system. However, many operating systems are designed to thwart priority inversion, using a mechanism known as "priority inheritance." Whenever a high-priority task attempts to enter a synchronized object that the operating system discovers is locked, the OS determines which task owns the lock on that object. If that object is owned by a task of lower priority, the low-priority task temporarily "inherits" the priority of the blocked task. (The OS actually boosts the low-priority task's priority level.) The boosted task now has a high enough priority that it can run, ultimately releasing the object and unblocking the high-priority task. The OS then returns the boosted tasks priority to what it had been, and priority inversion is averted.

**Priority inversion can occur in almost any multitasking system**

Unhappily, at this point, I am unaware of a JVM that includes this mechanism for handling priority inversion. Which means, of course, that it's up to programmers like you and me to wrestle with it. So what else is new?

### Code Online

All the code in this article is available for free at www.java-pro.com. Use the Locator+ codes shown below to download the code. Free online registration is required. See the Online Resources box in "Letters to the Editor" for more information.

***Inverted Priorities***
Registered Locator+: **JP0299**
Premier Locator+: **JP0299IT**

**Listing 2 Elevator Simulation Application** This listing shows the application class for the elevator simulation. Notice that in the StartElevator() method (which starts the elevator threads) the highest and lowest priority threads are started first. This lets you see how the simulation runs without priority inversion in effect. Then, the middle-priority elevator thread is started. Priority inversion (at least, on my NT system) hit almost immediately.

```
import java.awt.*;
import java.lang.Thread;

//
// Class to implement Elevator application for
// illustrating priority inversion.
//
public class ElevatorApp extends Frame
{
  public ElevatorApp()
  { setTitle("Elevator Thread Demo");
    c=new Canvas();
    add("Center",c);
  }

  public void StartElevator()
  {
    // Create a new BinarySemaphore object
    bs = new BinarySemaphore();

    // Start 1st and 3rd elevators
    e1 = new Elevator(c, 10, Thread.NORM_PRIORITY+2,
      Color.black, bs);
    e1.start();
    e3 = new Elevator(c, 90, Thread.MIN_PRIORITY,
      Color.blue, bs);
    e3.start();

    // Let them run unmolested for a few seconds
    try {
      Thread.sleep(8000);
    }
    catch (InterruptedException ex)
    { }

    // Start 2nd elevator
    e2 = new Elevator(c, 50, Thread.NORM_PRIORITY-1,
      Color.red, bs);
    e2.start();

  }

  public static void main(String[] args)
  {
    ElevatorApp f = new ElevatorApp();
    f.setSize(300,200);
    f.show();

    // Wait for screen to get drawn
    try {
      Thread.sleep(2000);
    }
    catch (InterruptedException ex)
    { }

    f.StartElevator();
    // Wait for a while, then end.
    try {
      Thread.sleep(50000);
    }
    catch (InterruptedException ex)
    { }
    System.exit(0);
  }

  private Canvas c;
  private Elevator e1, e2, e3;
  private BinarySemaphore bs;
}
```

# Headline News

## Saving state in a Web session becomes easy with cookies and the Servlet Session object

by Larry O'Brien

Larry is an independent trainer and consultant and developed *Bruce Eckel's Hands-On Java CD*. He is the chair of the "Internet and Intranet Development" track for the Software Development conference. Reach him at larryo@altnetinc.com.

Even the simplest dynamic site requires a huge commitment on the part of the publisher. Users might let you slide if you tell them "there's no news today," but they will not be as tolerant of seeing the same items under the "What's New" banner for months at a time. When creating a dynamic site, it is imperative to take full advantage of the power of the computer. It's hard enough to create the items for release without trying to mix them with the HTML necessary for a professional appearance. The servlets in this month's column will demonstrate the creation of a password-protected "What's New" page that's dynamically generated from a JDBC-accessed database, highlighting several of the crucial elements for a fully dynamic site.

If you want to create a "What's New" page that only shows the items that have changed since a user's last visit, you need a way to keep track of users and their last visit. In the lightweight world of the World Wide Web, where every hit on a Web server is stateless, this can take some doing. The cookie, originally introduced by Netscape, has been the best way to associate a particular user with hits on the server.

A cookie is a small piece of data sent down by the server as part of the HTTP response. If the browser supports cookies, subsequent HTTP requests to the provider are accompanied by the cookie as part of the request headers. Cutting cookie code can be crafty. Fortunately, the javax.servlet.http package encapsulates basic cookie functionality and allows you to manipulate things such as the cookie's lifespan with standard get/set accessor functions. Additionally, it is primarily by cookies that servlet servers provide arbitrary amounts of state information about a particular session.

You should be aware that a variety of security concerns might have prompted your users to disable their browser's cookie support. Certain browser defects (that were patched almost instantly) have allowed the possibility that hackers could read cookies from different domains, allowing them to determine that you've spent time at www.all_nude_sheep.com and subsequently derail your Senate confirmation hearings. At a more subjective level, the use of cookies is seen by some as inherently invasive. Cookies could be used to supply a site provider with extensive detail of your browsing session. Finally, a particularly stupid site provider might send sensitive information, like a credit-card number, in a cookie over a non-secure HTTP connection.

To get a better idea of the percentage of your site's users who have disabled their cookie functionality you could update the CapabilityServlet presented in this column in the November 1998 issue of *Java Pro*. Unfortunately, such users will be largely shut out of dynamic Web sites such as the one presented here.

### Implementing Login Functionality

We want to limit access to our Web site to registered users, therefore we need to implement login functionality. Because this is conceptually different from showing news items, and something that is likely to be useful in its own right, we'll place Login functionality in a separate servlet. But how do two separate servlets communicate? An example can be seen in Listing 1, which shows the WhatsNew servlet in its entirety.

The `WhatsNew.init()` method reads, as an initialization variable, the URL of the login servlet. The `init()` method also calls the `WhatsNewDatabase.getDatabase()` method, which returns a handle to an initialized database, as will be discussed later. The `WhatsNew.doGet()` method calls the `HttpServletRequest.getSession(boolean)` method to return an object unique to a particular browsing session. The method is called with the boolean value

**Listing 1 WhatsNew.java** In the "What's New" servlet users who have on logged on yet will be redirected to the login servlet.

```
package servletsolutions.whatsnew;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.text.*;

public class WhatsNew extends HttpServlet{
  WhatsNewDatabase newsDatabase;
  String loginURL;

  public void init(ServletConfig sc)
    throws ServletException{
    super.init(sc);
    loginURL = getInitParameter("loginServletURL");
    try{
      newsDatabase = WhatsNewDatabase.getDatabase();
    }catch(ClassNotFoundException ce){
      log(ce.toString());
      throw new UnavailableException(this,
        "Could Not Connect To Database");
    }
  }

  protected void doGet(HttpServletRequest req,
    HttpServletResponse res) throws ServletException{
    res.setContentType("text/html");
    try{
      PrintWriter out = res.getWriter();
      //Create session if not extant
      HttpSession session = req.getSession(true);
      if(session.isNew()){
        res.sendRedirect(loginURL);
      }else{
        out.write("<HTML><HEAD>");
        try{
          String user = (String) session.getValue("User");
          if(user == null){
            throw new Exception("Unknown User");
          }else{
            out.write("<TITLE>What's
              New</TITLE></HEAD><BODY>");
            String news = newsDatabase.getNews(user);
            out.write(news);
            SimpleDateFormat sdf =
              new SimpleDateFormat("MM/dd/yyyy");
            newsDatabase.updateLastUserVisit
              (user, sdf.format(new Date()));
            Cookie lastVisitCookie = new Cookie
              ("LastVisit", sdf.format(new Date()));
            lastVisitCookie.setMaxAge(60 * 60 * 24 * 365);
            res.addCookie(lastVisitCookie);
          }
        }catch(Exception e){
          log(e.toString());
          out.write("<TITLE>Exception in servlet</TITLE>");
          out.write("</HEAD><BODY>" + e.toString());
        }
        out.write("</BODY></HTML>");
        out.close();
      }
    }catch(IOException ioe){
      //No response writer, so hard to alert the user...
      log(ioe.toString());
      throw new ServletException
        ("Couldn't get PrintWriter for response");
    }
  }
}
```

set to true so that if a session has not been initialized one will be created. Having retrieved a Session, the method `session.isNew()` is called and, if the session is new, instead of continuing execution of the WhatsNew servlet, the `HttpServletResponse.sendRedirect(URL)` method is called with the login servlet's URL.

The Login servlet (see Listing 2) has an `init()` method almost identical to the WhatsNew servlet; it reads a URL for the WhatsNew servlet and uses the `WhatsNewDatabase.getDatabase()` method just like WhatsNew servlet did. The `doGet()` method of the login servlet is linear—it writes a simple login page. The Login page uses a form whose action is defined as causing a POST request to the login servlet itself. When the user presses the "Submit" button, their name and password are posted back to the login servlet, causing the `doPost()` method to fire.

The `doPost()` method uses the `javax.servlet.http.HttpServletRequest.getParameters()` method to return the values for "User" and "Password." `getParameters()` returns arrays, but we know that the login form presents just one value for each, so we assign the first element of the array to the userName and passWord variables. We then call the `WhatsNewDatabase.login(String user, String password)` method, which returns a boolean. If the login was unsuccessful, we output a simple error message, but if the login was successful, we use the `redirect()` method once again, this time sending the session back to the WhatsNew servlet. Before we do that, however, we use the `Session.putValue()` method to store the username of the login. `Session.putValue()` is a convenient way to store state information throughout a session. Behind the scenes, this state information may be stored in the form of a cookie or, if the user has shut off cookie functionality, through URL rewriting.

**Why use this double-checking technique rather than simply make the call to getDatabase() synchronized?**

When we reenter the WhatsNew servlet's `doGet()` method, the call to `session.isNew()` returned false, so instead of being redirected to the login screen again, we continue on in `WhatsNew.doGet()` (see Figure 1). We retrieve the username from the Session object, using `getValue()`, and pass that in to the WhatsNewDatabase's `getNews()` method, which returns an HTML-formatted list of news items. We output the information and create a cookie called "LastVisit" based on the current time. Once created, the cookie is sent to the user with a call to the HttpServletRequest's `addCookie()` method.

Finally, we catch various exceptions, log them, and attempt to write a diagnostic to the HTML output as well. The login and WhatsNew servlets, the Session object, and

the "LastVisit" cookie work in combination to give us basic dynamic capabilities.

## Connecting to the Database

The actual work of writing the pages is deferred to another class—the `WhatsNewDatabase` class. As I discussed in January's Servlet Solutions column, I am of the opinion that encapsulating database access in a single class is appropriate in many situations. (James Cooper's December Javatecture column, "Handling Databases More Effectively" discusses a similar design.)

As discussed previously, the two servlets do not instantiate the `WhatsNewDatabase` but use the static method `WhatsNewDatabase.getDatabase()` to return a handle to the database. If you look at `WhatsNewDatabase` in Listing 3, you can see that I've used the Singleton design pattern so that one-and-only-one instance of the `WhatsNewDatabase` can be created. The Java implementation of the Singleton design pattern uses static fields (such as the JDBC driver name, the database name, and the user and password fields), a static handle to the Singleton class itself, and a constructor declared as private, making it impossible for external classes to instantiate an instance of the class.

Because the `WhatsNewDatabase` constructor is private, the only way to instantiate a `WhatsNewDatabase` instance is from within a static method. In many situations, when implementing a Java Singleton, you can use a simple static initializer or a static initialization block to do the singleton instantiation. However, in this case, our initialization is a little more complex than can be comfortably handled within a static initialization block, so I have opted to use "lazy" initialization instead. With this strategy, the static handle to the `WhatsNewDatabase` is initialized to null. When `getDatabase()` is called, the handle is checked to see if it's null or not. If not, it returns a handle to the initialized database. But the first time `getDatabase()` is called, the handle *will* be null, and so we'll lazily initialize the database.

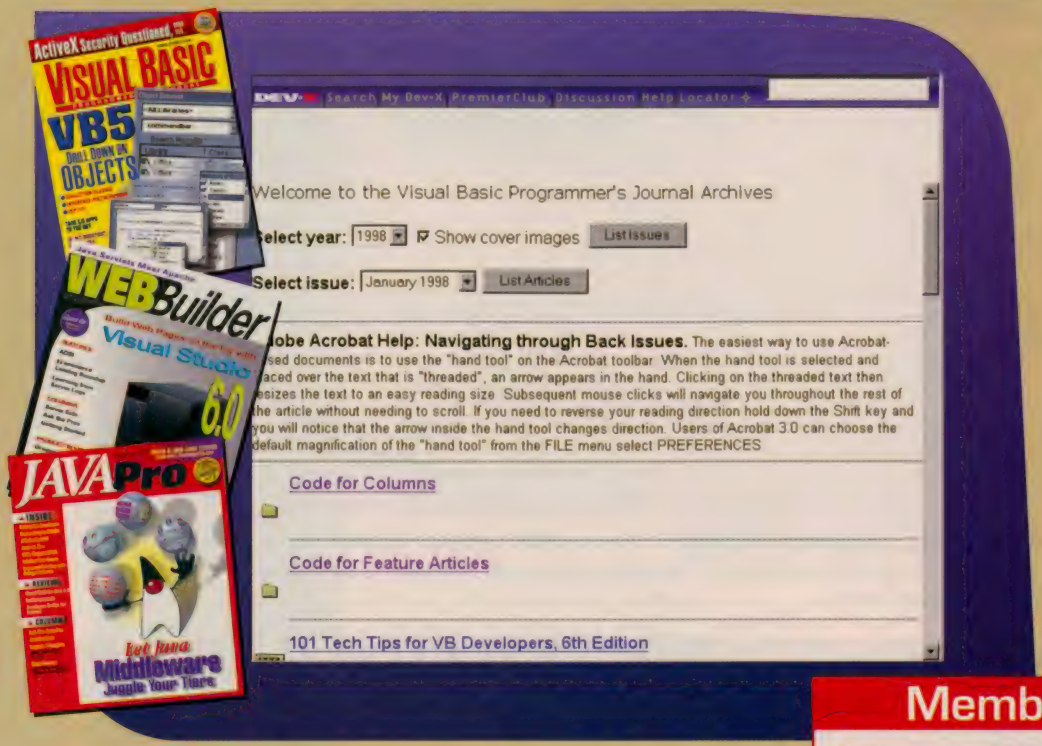
Notice that after the check to see if the `WhatsNewDatabase` is null, we synchronize on the `WhatsNewDatabase` Class object. This ensures that

**Listing ② Login.java** The login servlet first creates a form and then checks the form's data against the database.

```
package servletsolutions.whatsnew;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class Login extends HttpServlet{
  WhatsNewDatabase loginDatabase;
  String whatsnewURL;

  public void init(ServletConfig sc)
    throws ServletException{
    super.init(sc);
    whatsnewURL = getInitParameter("whatsnewServletURL");
    try{
      loginDatabase = WhatsNewDatabase.getDatabase();
    }catch(ClassNotFoundException ce){
      log(ce.toString());
      throw new UnavailableException(this,
        "Could Not Connect To Database");
    }
  }

  protected void doGet(HttpServletRequest req,
    HttpServletResponse resp) throws ServletException{
    String responseHTML = new String();
    responseHTML = doLoginScreen();
    resp.setContentType("text/html");
    try{
      PrintWriter out = resp.getWriter();
      out.write(responseHTML);
      out.close();
    }catch(IOException e){
      log(e.toString());
      System.err.println(e);
    }
  }

  protected void doPost(HttpServletRequest req,
    HttpServletResponse resp) throws ServletException{
    String userNames[] = req.getParameterValues("User");
    String userName = userNames[0];
    String passWords[] = req.getParameterValues("Password");
    String passWord = passWords[0];

    resp.setContentType("text/html");

    boolean loginSuccess = loginDatabase.login
      (userName, passWord);
    if(loginSuccess){
      HttpSession session = req.getSession(true);
      session.putValue("User",userName);
      try{
        resp.sendRedirect(whatsnewURL);
      }catch(IOException e){
        try{
          PrintWriter out = resp.getWriter();
          out.write("<HTML><HEAD><TITLE>Login
            successful</TITLE></HEAD>");
          out.write("<BODY>You may now <a href=\""
            + whatsnewURL + "\">");
          out.write("proceed to the whatsnew
            servlet</a></BODY></HTML>");
          out.close();
        }catch(IOException ex){
          log(ex.toString());
          System.err.println(ex);
        }
      }
    }else{
      try{
        PrintWriter out = resp.getWriter();
        out.write("<HTML><HEAD><TITLE>Login
          unsuccessful</TITLE></HEAD>");
        out.write("<BODY>");
        out.write("<p>The user name and password you supplied
          were not recognized. </p>");
        out.write("<p>Use your \"Back\" button to try
          again.</p></BODY></HTML>");
      }catch(IOException e){
        log(e.toString());
        System.err.println(e);
      }
    }
  }

  protected String doLoginScreen(){
    StringBuffer resp = new
      StringBuffer("<HTML><HEAD><TITLE>Login
      </TITLE></HEAD><BODY>");
    resp.append("<form method=\"post\"
      action=\"/servlet/login\">");
    resp.append("<p>User Name</p><p>");
    resp.append("<input type=\"text\" name=\"User\"
      maxlength=\"50\" size=\"50\">");
      resp.append("</p><p>Password</p><p>");
    resp.append("<input type=\"password\" name=\"Password\"
      size=\"50\">");
    resp.append("</p><p>");
    resp.append("<input type=\"submit\" name=\"submit\"
      value=\"submit\">");
    resp.append("</p>");
    resp.append("</form></body></html>");
    return resp.toString();
  }
}
```

**Attention Developers**

# Search Articles. Download Code.


ActiveX Security Questioned
VISUAL BASIC
VB5
Dive Down on Objects
WEB Builder
Visual Studio 6.0
JAVA Pro
Let Java Middleware Juggle Your Data
Search My Dev-X PremierClub Discussion Help Locator
Welcome to the Visual Basic Programmer's Journal Archives
Select year: 1998 Show cover images List Issues
Select issue: January 1998 List Articles
Adobe Acrobat Help: Navigating through Back Issues. The easiest way to use Acrobat-based documents is to use the "hand tool" on the Acrobat toolbar. When the hand tool is selected and placed over the text that is "threaded", an arrow appears in the hand. Clicking on the threaded text then resizes the text to an easy reading size. Subsequent mouse clicks will navigate you throughout the rest of the article without needing to scroll. If you need to reverse your reading direction hold down the Shift key and you will notice that the arrow inside the hand tool changes direction. Users of Acrobat 3.0 can choose the default magnification of the "hand tool" from the FILE menu select PREFERENCES
Code for Columns
Code for Feature Articles
101 Tech Tips for VB Developers, 6th Edition

PREMIER Club

Join now and Premier Club membership is **ONLY $20** for six months. You'll get an additional 25% off if you already subscribe to *VBPJ*, *WEB Builder* or *Java™ Pro*.

## Member Benefits

- **Access to All Previous Issues.** Over 1,000 articles from *Visual Basic Programmer's Journal*, *WEB Builder*, Avatar, *Java™ Pro*, and *Getting Started with Visual Basic*.
- **Code!** Members gain admission to the entire archive area of DevX, with complete searchable text from all these publications.
- **Even More Code.** Extra apps with complete source, plus special utilities.
- **Email Product Reviews.** Get e-mail issues of Visual Programming++ (VP++), with product reviews of the latest development tools as they are released.
- **Tools and Links.** Unlimited access to DevX's searchable product database, the only comprehensive, online listing of every significant programming tool on the market.
- **Club Privileges.** Special discounts, discussions, services, and much more.

**The Development Exchange's (DevX's) Premier Club** offers you access to the largest resource of programming information and tools available anywhere. Your Premier Club membership starts with immediate, unlimited access to information and downloadable code from the entire archive of FTP's leading programming titles—information that will save you tons of coding time.

**And as a Premier Club member**, your access extends to any of the unique development zones and sites within the DevX family of web sites—all with related articles, key tools, and dedicated programming information on VB, C++, Java, and Enterprise development.

**To start your membership**, point your browser to DevX's Premier Club area, and follow the instructions.

## Enroll Today at

# www.premier-club.com

DEV·X The Development Exchange

www.devx.com

Microsoft, Windows, Visual Basic, Visual C++ are registered trademarks of Microsoft Corporation. Java is a registered trademark of Sun Microsystems.
The Development Exchange, DevX, CareerLink, and Ask the Pros are trademarks of Fawcette Technical Publications.
Fawcette Technical Publications, 209 Hamilton Avenue, Palo Alto, CA 94301-2500. Phone: 650.833.7100; Fax: 650.853.0230; www.devx.com.

if two calls to `getDatabase()` are happening concurrently and both see that the handle is null, only one can move forward. The line after the synchronization call repeats the check to see if the handle to the `WhatsNewDatabase` is still null. For the first thread that acquires the synchronization lock this will naturally still be true. That thread then calls the private constructor and assigns the resulting handle to the Singleton handle. The first thread then exits the synchronization block, allowing the second thread to enter. This time, the second call to `if(singleton == null)`

**The Singleton pattern is quite common in Java programming**

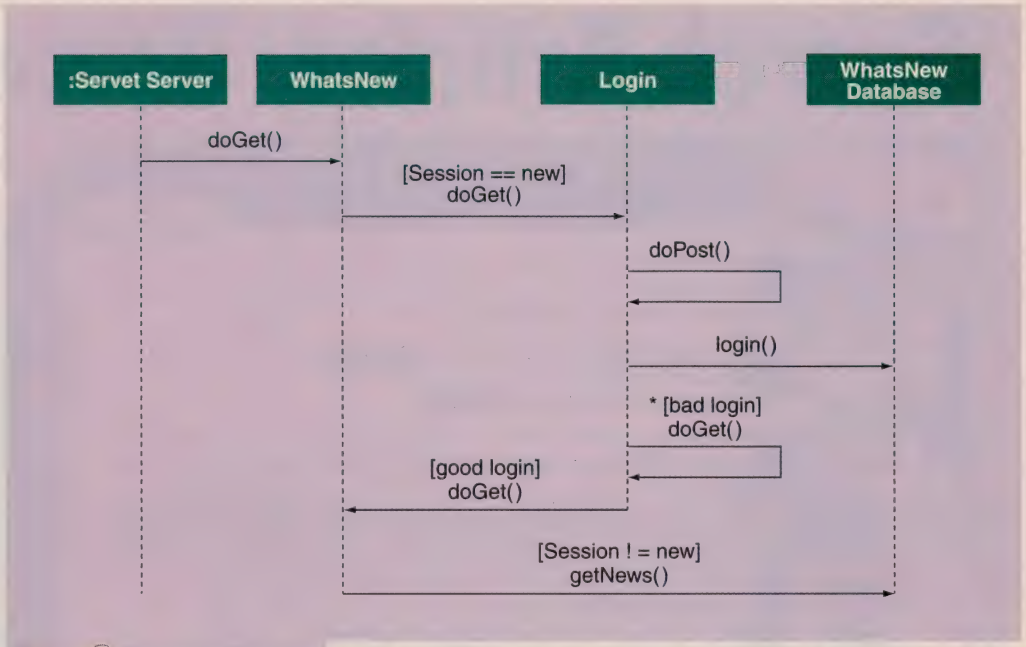



**Figure 1 Sequence Diagram** In this simplified sequence diagram, the WhatsNew servlet redirects to the Login servlet as necessary. The login servlet loops until login is successful, when it redirects to the WhatsNew servlet.

**Listing 3 The Behind-the-Scenes Work** This class encapsulates all database functionality, including initialization, data access and updating, as well as properly formatting the output.

```
package servletsolutions.whatsnew;

import java.sql.*;

class WhatsNewDatabase{
    static final String driverName =
      "sun.jdbc.odbc.JdbcOdbcDriver";
    static final String dbName =
      "jdbc:odbc:news";
    static final String user = "";
    static final String pass = "";

    static WhatsNewDatabase singleton;
    //Can't use static initializer for singleton because
    //constructor throws exceptions

    private WhatsNewDatabase() throws
    ClassNotFoundException{
      Class.forName(driverName);
    }

    static WhatsNewDatabase getDatabase() throws
      ClassNotFoundException{
      if(singleton == null){
        synchronized(WhatsNewDatabase.class){
          if(singleton == null){
            singleton = new WhatsNewDatabase();
          }
        }
      }
      return singleton;
    }

    String getNews(String user){
      try{
        Connection cnctn =
          DriverManager.getConnection(dbName, user, pass);
        Statement s = cnctn.createStatement();
        ResultSet r = s.executeQuery(
        "SELECT Headlines.*, Users.UserName, Users.ID " +
        "FROM Headlines, Users INNER JOIN Subscriptions ON"
          +
          " Users.ID = Subscriptions.UserID WHERE " +
          "(((Users.UserName)= '" + user + "') AND " +
          "((Headlines.Posted) > " +
          "[Subscriptions].[RetrieveNewsSince])" +
          "AND ((Headlines.Category)=2) AND " +
          "((Subscriptions.Category)=2) AND " +
          "((Subscriptions.UserID)=[Users].[ID]));");
        StringBuffer news = new StringBuffer();
        if(r != null){
          while(r.next()){
            news.append("<hr><p>" + r.getString("Posted"));
            String targetURL = r.getString("targetURL");
            if(targetURL != null){
              news.append(" <a href=\"");
              news.append(targetURL);
              news.append("\">");
            }
            news.append(" " + r.getString("Head"));
            if(targetURL != null)
              news.append("</a>");
            news.append("<p>" + r.getString("Abstract"));
            news.append("</p>");
          }
        }
        return news.toString();
      }catch(SQLException se){
        return "Couldn't connect to database.
          Try again later.<p>" + se.toString();
      }
    }

    boolean login(String user, String pass){
      try{
        Connection cnctn =
          DriverManager.getConnection(dbName);
        Statement s = cnctn.createStatement();
        ResultSet r = s.executeQuery
          ("SELECT * FROM USERS WHERE "+
        "[UserName] = '" + user + "' AND [Password] = '"
          + pass + "';");
```

*Listing continued on next page.*

# www.cplus-zone.com

# Need C++ Programming Information *Now*?

**It's all here on DevX's Cplus-zone—the most comprehensive independent source of technical information for Windows and interactive developers.**

Join today and get immediate access to powerful tools, valuable code and DevX's vast resource of C++ programming information, including:

- **Technical articles and code** from the pages of *Visual Basic Programmer's Journal* and *WEB Builder* magazines
- **More Downloadable Code**—the best shareware and product demos to save you weeks of surfing!
- **Daily C++ programming tips**—plus a library of thousands more
- **Products**—find the right tool from a database of over 5,000 tools
- **Reviews of 100's of products**—with new reviews posted weekly
- **Dedicated C++ Development forums**
- **Buy Programming Books Online** Save up to 40% on selected titles, plus get editors' picks and reviews of top C++ titles

Cplus-Zone is a member of the family of programming sites located on The Development Exchange at http://www.devx.com

DEV·X The Development Exchange

Windows, Visual Basic, and Microsoft are registered trademarks of Microsoft Corporation. The Development Exchange and DevX are trademarks of Fawcette Technical Publications.
Fawcette Technical Publications, 209 Hamilton Avenue, Palo Alto, CA 94301-2500. Phone: 650.833.7100; Fax: 650.853.0230; www.devx.com

## Advertiser Index


**Caribou Lake Software** 2
1-612-921-8338
http://www.CaribouLake.com

**Compuware, NuMega** 13
1-800-4-NUMEGA
http://www.numega.com

**DevX-C++-Zone** 75
1-800-848-5523
http://www.Cplus-zone.com

**DevX** 27
1-800-848-5523
http://www.devx.com

**DevX-Java-Zone** 57
1-800-848-5523
http://www.java-zone.com

**DevX-Premier Club** 73
1-800-848-5523
http://www.premier-club.com

**Distinct Software** C3
http://www.distinct.com

**Enterprise Development 2000** 43
1-800-848-5523
http://www.enterprisedev.com

**Enterprise Development** 33
Magazine Subscriptions
1-800-848-5523
http://www.devx.com

**EnterpriseSoft** 4
http://www.EnterpriseSoft.com

**FTP Books** 45
1-800-848-5523
http://www.devx.com

**IBM Corporation** 39
http://www.software.ibm.com
/websoftware

**ILOG** 11
1-800-FOR-ILOG
http://www.ilog.com

**Inquiry.com** 51
1-800-848-5523
http://www.inquiry.com

**Java Pro** 65
Magazine Subscriptions
1-800-848-5523
http://www.devx.com

**KL Group** C2 & 1
1-800-663-4723
http://www.klg.com

**MapInfo Corporation** 23
1-800-327-8627
http://www.mapinfo.com

**Microsoft Corporation** 19
http://msdn.microsoft.com/visualj

**MindQ Publishing** 17
1-800-646-3008
http://www.mindq.com

**ObjectShare** 8
1-800-973-4777
http://www.objectshare.com

**ObjectSoft Inc.** 6
1-888-OBJSOFT
http://www.objsoft.com

**ProtoView** C4
Development Corporation
1-800-231-8588
http://www.protoview.com/java

**Tidestone Technologies** 15
1-800-884-8665
http://www.tidestone.com


### Showcase


**Caribou Lake Software** 76
1-612-921-8338
http://www.CaribouLake.com

**Data Representations** 76
1-888-307-9550
http://www.datarepresentations.com/

**ErgoTech Systems Inc.** 76
1-888-374-6832
http://www.ergotech.com


### Product Information

- To get immediate details on products advertised in this issue, you can simply log onto our online services at **http://infoasap.devx.com**.
- This provides a list of direct links to this month's advertiser's Web sites.

### Advertising Sales Contacts


- **Kent Wu, Advertising Director**
  650-833-7127; kwu@fawcette.com
- **Regan Swett, Regional Sales Manager**
  650-833-7154; rswett@fawcette.com
- **Kelly Keane, Western Regional Sales Manager**
  650-833-7128; kkeane@fawcette.com
- **Shana Deane, Eastern Regional Sales Manager**
  413-586-6074; sdeane@fawcette.com


This listing is provided as a courtesy to our readers and advertisers. The publisher assumes no responsibility for errors or omissions.
**Sales Department:** 209 Hamilton Avenue • Palo Alto, CA, USA 94301-2500 • 650-833-7100


JAVAPro Product Showcase

www.java-zone.com

**Java Solutions Delivered Daily**

Jump on today and get immediate access to the most comprehensive source of expert information and code to help you develop serious Java applications, including:

- Technical articles and code
- Daily programming tips
- Download Java tools
- Search Products
- Product Reviews
- Dedicated Java Forums
- 40% off Books of the Week

DEVX The Development Exchange

Fawcette Technical Publications
209 Hamilton Avenue • Palo Alto, CA 94301-2500
Phone: 650.833.7100 • Fax: 650.853.0230
www.devx.com

Java is a registered trademark of Sun Microsystems, Inc.
DevX is a trademark of Fawcette Technical Publications.

http://www.devx.com

**JSQL™**

FOR THOSE SERIOUS ABOUT JAVA DEVELOPMENT

Recently named to the Information Week's Top 100 Internet and Intranet Solution Series (10/5/98 Issue), JSQL™ provides the middleware you need to develop secure, robust, scalable, high performance bridges between Java and your Enterprise applications, databases and data warehouses.

CARIBOU LAKE SOFTWARE

"Superior Java/RDBMS/DW Products and Services"

8400 Normandale Lake Blvd., #920
Minneapolis, MN 55437
Tel: 612 921-8330 Fax: 612 921-8338

http://www.CaribouLake.com

**FREE DOWNLOAD!**

The Simplest Way to Create Java...

Simplicity for Java™

- The Code Sourcerer™ writes Java code for you
- Executes any Java code on-the-fly without compiling
- Supports all AWT & Swing components
- Import any JavaBeans™

http://www.datarepresentations.com

**Virtual Instrumentation Beans™**

Build dynamic real-time virtual instrumentation panels with Java™ -- view live data anywhere!

ErgoTech Systems, Inc.
(888) ERGOTECH (505) 662-5156

http://www.ergotech.com

*Listing continued from previous page.*

```
        return r.next();
      }catch(SQLException se){
        System.err.println("Login error");
        System.err.println(se);
        return false;
      }
    }

    protected void updateLastUserVisit
      (String userName, String lastVisit){
      try{
        Connection cnctn =
          DriverManager.getConnection(dbName);
        Statement s = cnctn.createStatement();
        String query = "UPDATE Users INNER JOIN
          Subscriptions ";
        query += "ON Users.ID = Subscriptions.UserID SET
          Subscriptions.RetrieveNewsSince = #";
        query += lastVisit;
        query += "# WHERE (((Users.UserName)='";
        query += userName;
        query += "') AND
          ((Subscriptions.UserID)=[Users].[ID]) AND
          ((Subscriptions.Category)=2));";
System.out.println(query);
        s.executeUpdate(query);
      }catch(SQLException se){
        System.err.println("Error updating user " +
          userName + " with " + lastVisit);
        System.err.println(se);
      }
    }
```

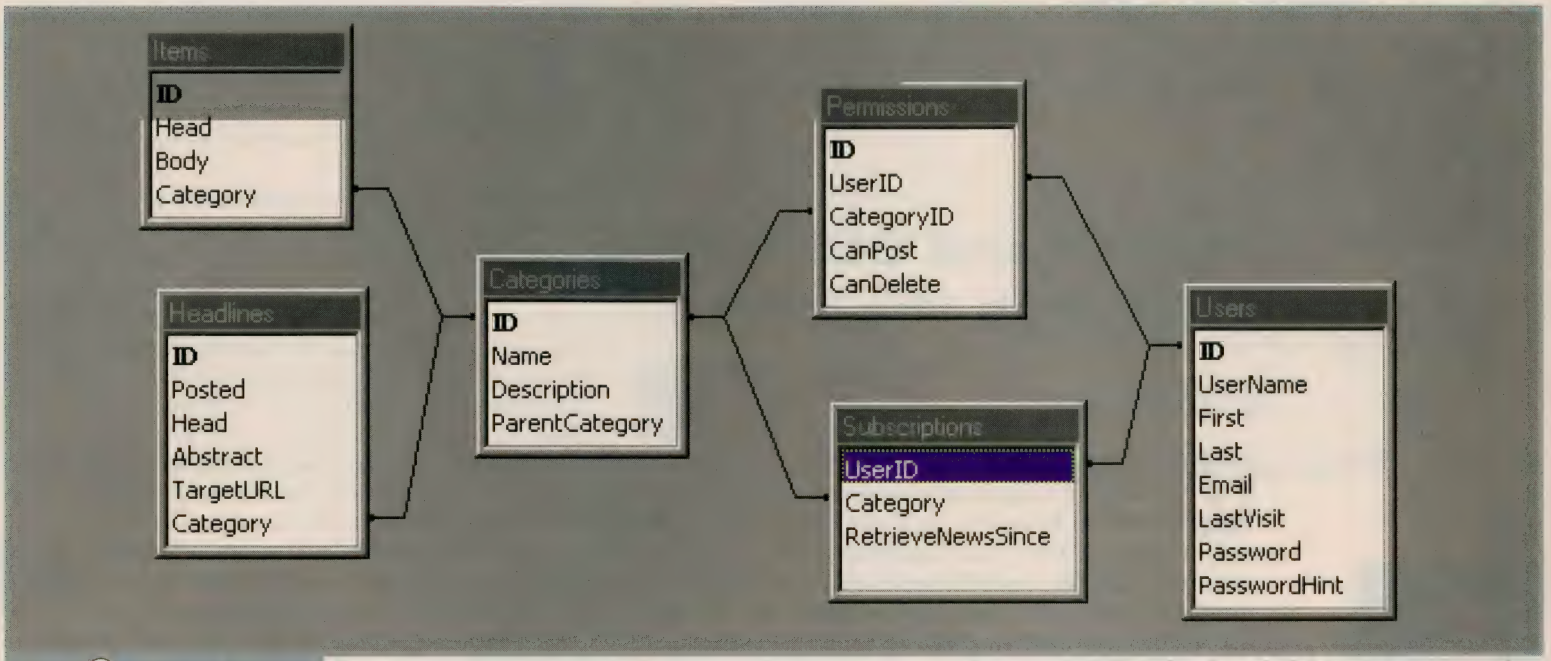



**Figure 2 Database Structure** Does this database structure look more complex than is necessary for a simple "What's New" servlet? It is—but stay tuned for future developments.

will fail (because the first thread just initialized it), and the second thread will not re-initialize the database.

Why use this double-checking technique rather than simply making the call to `getDatabase()` synchronized? If the method were declared synchronized, no two threads could enter it simultaneously and we could do with a single call to `if(singleton == null)`. The answer lies in the relative rarity of the thread conflict. Thread contention occurs only when the first two calls to the method are concurrent. Once the database is initialized, we never have to worry about thread safety in this method again. However, if we synchronized the `getDatabase()` method call, we would pay the synchronization overhead penalty every time the method was called, whether it was the first call or the ten-thousandth. In our small servlet example, the difference is insignificant, but it's good to know the technique, because the Singleton pattern is quite common in Java programming.

In addition to initialization, the `WhatsNewDatabase` class encapsulates the various database queries we want—`login(String username, String password)` is defined as returning a boolean; `getNews(String user)` returns an HTML-formatted set of news items from the database; and `updateLastUserVisit(String userName, String lastVisit)` stores in the database the fact that the user has visited the site (remember that this is also "remembered" by the user's browser in the form of a cookie).

**The answer lies in the relative rarity of the thread conflict**

If you were to examine the SQL calls in the various methods, you'd see evidence of a database structure considerably more complex than necessary to create a simple news page (Figure 2). You can see that a User has a one-to-many relationship with a table called Subscriptions, and that Headlines have a many-to-one relationship with a table called Categories. What's the purpose of all this complexity? How about making your site a portal? The details as to how will have to wait for a later column. ■

## Code Online

All the code in this article is available for free at www.java-pro.com. Use the Locator+ codes shown below to download. Free online registration is required. See the Online Resources box in "Letters to the Editor" for more information.

**Headline News**
Registered Locator+: **JP0299**
Premier Locator+: **JP0299SS**

# Platform Independence Calls for Platform Savvy

## Thread and GUI coding put WORA to the test

by Paul Tyma

Paul Tyma is president and chief scientist at preEmptive Solutions Inc., a Java training and consulting firm. Paul is the lead author of the book *Java Primer Plus* and a frequent conference speaker on Java Performance. He is currently a Ph.D. candidate in computer engineering at Syracuse University and lead architect for preEmptive's DashO Java bytecode optimizer/obfuscator. Reach him at ptyma@preemptive.com.

Users can be self centered. They're not particularly interested in applications that are object oriented, platform independent, or otherwise. They want to get their work done efficiently, period. If a given application facilitates that, that's the application they'll use. Aside from a few Commodore diehards, few users have more than one architecture on their desks. As far as most users are concerned, if their applications run, they're happy. If it happens to run on scads of other architectures and operating systems, that's just spiffy—but of little importance.

It's the programmer's job to leverage platform independence to please the most number of users possible. In past columns, we've looked at several different runtimes and tested their abilities. It's nice to know which is the fastest runtime, but it's more important that your program runs correctly, fast or otherwise. The challenge is that every runtime has its idiosyncrasies.

I've found that a notable portion of my company's business is focused on making applets work in both Internet Explorer (IE) and Netscape Navigator. It ought to be simple enough, but anyone who's tried to accommodate both knows that it's a delicate task. The behavioral differences between IE and Navigator are sometimes so subtle, so ingenious, and so without reason that they seem intentionally woven into the runtime's fabric. Hmmm. But don't start pointing fingers, both IE and Navigator runtimes have their problems, and if you straddle the two architectural bounds, you open up a whole new can of worms.

In some respects, this is a treatise on where the ubiquitous idea of platform-independence all falls apart. Sort of a horrible thought. The good news is, understanding the differences in runtimes and even platforms can give you a head start in making your code work. You can't blindly rely on the promise of platform independence. You need to prepare your code to work on your user's platform. If you plan before you code, it will be part of your design. If you do it afterwards it will be a "workaround."

### GUI, GUI, Gone

GUIs and threads are the two most troublesome elements of platform independence, and most Java programmers would probably scream that GUI programming has far more problems. A perfectly good Windows GUI can look like a bugger on a Macintosh. The subtle nuances across VMs are endless. GUIs have many problems—or, more objectively, many idiosyncrasies across runtimes.

Unfortunately, the idiosyncrasies are too numerous and too diverse to completely cover here. Of course, the JDK on Windows tends to be viewed as a relative standard. JITs from Netscape and Symantec also follow the rules quite well, but Microsoft's Jview has some significant issues, especially with Swing. Once you move to new architectures, GUIs tend to be less predictable, JDK or otherwise.

The best defense is plenty of early testing and checking the Usenet for folks with the same problems as you (they *do* exist). Test against multiple runtimes. If you develop and test on only one browser, you are sure to have an application that works on only one browser.

### Thread Traits

The clearest difference between runtimes is the threading architecture of the underlying OS, which either supports preemptive or non-preemptive multithreading. Multi-

threaded design in your programs is arguably as important as object-oriented design. The concepts work hand in hand to create highly reusable, robust, and responsive applications.

The type of multithreading greatly impacts behavior. With preemptive multithreading, the operating system can mercilessly interrupt a thread and execute another, and with non-preemptive multithreading, the OS must wait until a thread voluntarily gives up before the CPU can execute another. The concepts call for opposing programming structures. Unfortunately, in Java you must program somewhere in the middle, taking extra steps to be sure you code runs on both types.

The key is to ensure that no thread ever has a stranglehold on the CPU. A program that runs fine on a Windows box could completely deadlock in Linux. Use `Thread.yield()` or `Thread.sleep()` liberally throughout intensive, yet preemptable, portions of your code.

To help you deal with the some of the runtime idiosyncrasies I've compiled this list of eight tips:

### tip 1 Speed Netscape's Applet Startup

A lot of people, distraught with the size of a download, will put the entire jar on their client machines. However, they continue to encounter the dreaded "Starting Java..." message before the applet starts. Try uncompressing that jar on the client side—you'll be amazed how fast the applet starts up. Also, if you can't put things on the client side, Java Application Packaging programs grab the minimum required working set out of the jars (classes and methods) and deliver just what your applet needs, reducing download and startup times.

### tip 2 Avoid Netscape's Overzealous Start/Stop

There's an amazing bug/feature in Netscape that calls start and stop each time you resize the browser. This is particularly nasty if you're trying to detect when someone actually leaves a Web page. It becomes difficult to determine if they left the page or just resized, because stop is called in both cases. There's no good Java solution. I've seen developers use a time-out thread to guess that the page had been exited (nearing the pinnacle of "hope-it'll-work" programming). The best solution uses JavaScript, exploiting its `onload` and `onunload` facilities. It can call into your applet doing whatever you need getting done. Strict IE applets can rely on start and stop for this functionality, but will also work with the JavaScript solution.

### tip 3 Prevent Java Applications from Taking Over

Some CPU intensive applications "take over" a Windows 95 machine. They are so processor intensive (and Windows 95 manages them so poorly) that the whole operating system comes to virtual halt. It's hard to even click anything once the application starts running. This problem isn't as obvious in Unix (CPU pegged to 100% is hardly noticeable from a user-interaction standpoint). But sprinkling a few `Thread.yield()` calls in your tight loops will give the unwary Windows 95 user much of his or her machine back while your app is running. This negligibly affects performance for other OSs.

### tip 4 Use IE as a Verifier

As someone who rips apart and reconstructs class files for a living, I can honestly say that IE is my best friend. Its verifier catches errors in class files better than any other verifier I've used. Want to check if that obfuscater is generating legal code? Send it through IE.

### tip 5 On Windows, Use Jview

I'm amazed at how few people realize Jview comes as part of Microsoft's operating system. It's the best performing JIT we've reviewed. It's like some sort of best-kept secret that Microsoft didn't market very well. (That sounds weird doesn't it?)

### tip 6 Get Line-Numbered Stack Traces

As one of the few exceptions to the previous tip, use the JDK (`-nojit` option for 1.1.7) when debugging an exception. If debugging info is stored in your classes you'll see exactly what line number caused the exception. JITs (at least those from Symantec and Microsoft) do not show line numbers in stack traces.

**What could be better than running one runtime regardless of the browser?**

### tip 7 Get the Plug-in

Browser runtimes giving you headaches? Get the Java Plug-in. What could be better than running one runtime regardless of the browser? Although the Java Plug-in can't do everything, it will put things on a level playing field and might solve your problems. Get the free download from that all-knowing Java site, http://java.sun.com/products/plugin/temp/index.html. Starting development using the Plug-in could save you hours and even days of work (trying to port to it later is more difficult).

### tip 8 Performance Problems? Run from java.util.*

Be afraid. Be very afraid. Although the java.util.Vector and java.util.Hashtable represent well-designed, reusable classes, they are, unfortunately, almost too reusable. They were designed to handle any use thrown at them. Many methods are synchronized and (necessarily) written quite generically. If you use these classes a lot, a tailored version to suit your needs can repay you with a noticeable speed up.

### Dealing with Diversity

The bad news is people are still writing new runtimes and the old ones are adding new features. Although that might sound like a good thing, it lessens the chances of true boilerplate runtime behavior. GUI tweaking has become a fine art that no one really wanted to learn, and it is sure to be a growing pain Java developers will be dealing with for a long time to come. ■

# Meet *FTP*

# *Fawcette Technical Publications*

This issue marks the eighth anniversary of Fawcette Technical Publications, which brings you print publications such as *Java Pro, Visual Basic Programmer's Journal, Enterprise Development,* and *Getting Started with Visual Basic;* The Development Exchange (DevX) family of Web sites, including Ask the Pros and *Web Builder* magazine; the VBCD and the Web Builder CD; and conferences such as Visual Basic Insiders' Technical Summit (VBITS), Enterprise Development 2000, NT Enterprise Developer's Conference, and VB Jump Start. We look forward to an exciting future developing products that serve you.

**FIRST ROW from bottom (L to R):**
Raylene Leavitt, HR/Executive Administrator
Ben Waltzer, DevX Advertising Coordinator
Betsy Slattery, *VBPJ* Associate Editor
Diane Matcheck, DevX Reviews Editor
Amanda Santos, *Enterprise Development* Editorial Assistant
Kelly Keane, *Java Pro* Regional Sales Manager
Tyler Sperry, *Java Pro* Editor in Chief
Jennifer Brucker, Conference Planner
Daniel Hernandez, Receptionist
Terrilyn Chance, *VBPJ* Associate Art Director

**SECOND ROW from bottom (L to R):**
Wendy Hartman-Carr, Controller
Nina Goldschlager, *VBPJ* Managing Editor
Regan Swett, *Java Pro* Regional Sales Manager
Elise Peterson, *Java Pro* Associate Editor
Kristin Fladager, Marketing Specialist
Kim Le, Senior Designer
Lisa Sidlow, *VBPJ* Regional Account Manager
Lara Reed, Catalog Manager
Nicole Pizzurro, *VBPJ* Associate Managing Editor
Ha Hoang, IS Specialist
Susan LaCroix, *VBPJ* Production Coordinator
Cecile Pendleton, *Enterprise Development* Account Manager
Kay Keppler, DevX Senior Editor
Chris Povio, *Enterprise Development* Regional Sales Manager

**THIRD ROW from bottom (L to R):**
Tara Brennan, *VBPJ* Sales/Production Assistant
Robyn Wandzell, Electronic Production Associate
Amy René Pelto, Design Production Associate
Kathy Sweeney, Production Artist
Tena Carter, General Manager, Conferences
Sam Azzouni, Accounts Receivable Accountant
Chris Ortiz, Customer Service Representative
James Fawcette, President
Kimberly Bell, Electronic Products Associate Art Director
Michael Furukawa, Associate Production Manager
Dan Reback, Regional Account Manager
Melissa Ramirez, Customer Service Representative
Daryl Haggard, Conference Planner
Libby Dickinson, Marketing Communications Manager
Karen Koenen, Circulation Director
Katie Hathaway, Associate Circulation Manager

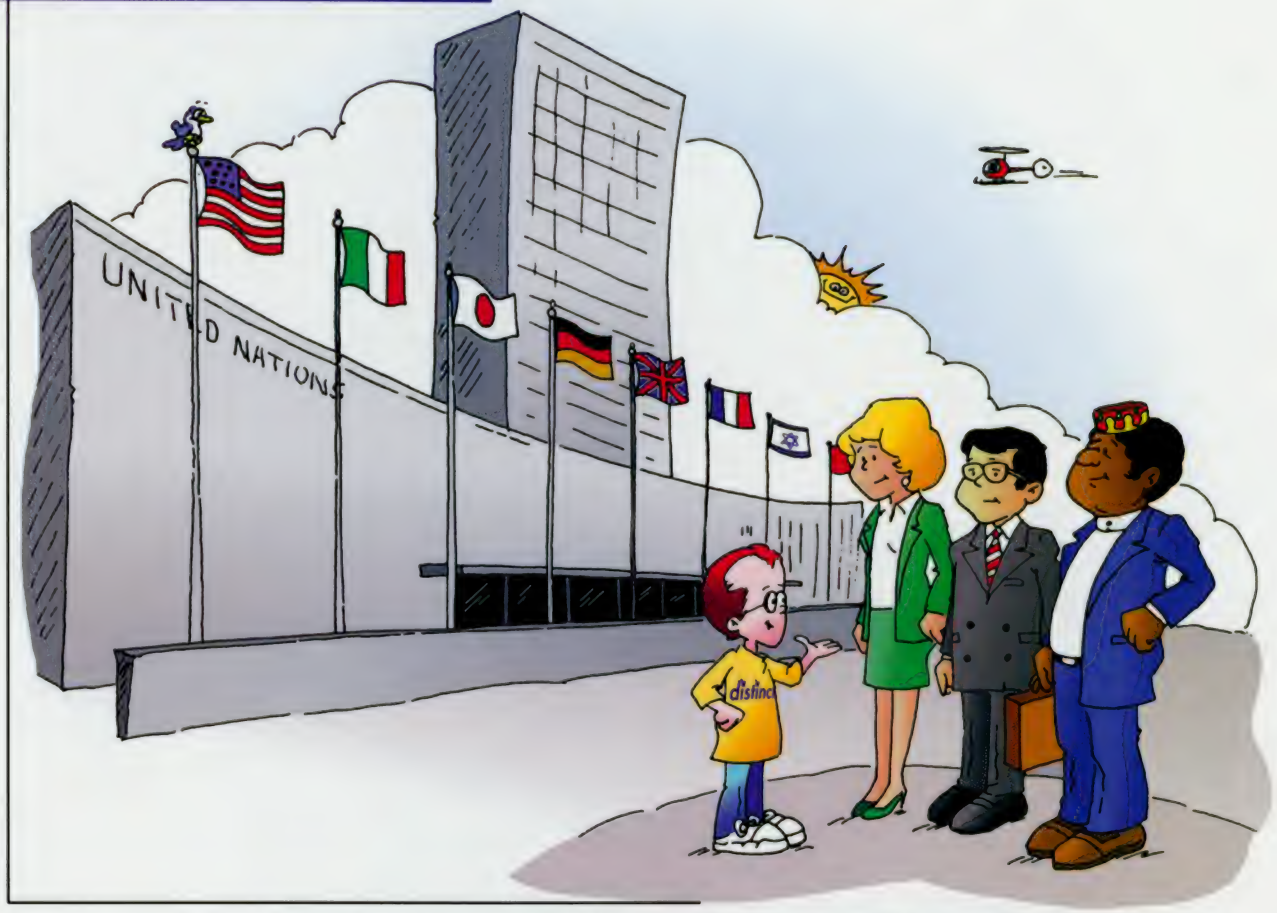
**FOURTH ROW from bottom (L to R):**
John Dorsey, *Java Pro* Managing Editor
Clary Alward, *VBPJ* Associate Managing Editor
Susannah Pfalzer, *VBPJ* Senior Editor
Sara Smith, *Enterprise Development* Associate Managing Editor
Sylvia Zele, DevX Associate Editor
Jeff Hadfield, *VBPJ* Editor in Chief
Timothy Haight, Editorial Director, Conferences
Matthew Carter, DevX Editor and Producer
Donna Ryan, Marketing Specialist
Bill Poulin, Web Developer
Rachel Palmer, *Enterprise Development* Production Coordinator
Michelle Grossman, Customer Service Representative
Jeff Miller, Vice President of Publishing
Julianne Olmstead, Ancillary Products Manager
Sherry Thomas-Zon, Marketing Director

**FIFTH ROW from bottom (L to R):**
Damian Zusi, Conference Sales Manager
Janet Brandt, Conference Planner
Evan deRouin, *VBPJ* Regional Account Manager
Trisha Merriam, Assistant to the Publisher
Richard Sims, *VBPJ* Advertising Director
James Kattinge, Senior Designer
Don Dugdale, *VBPJ* Senior Editor
Lee Thé, *Enterprise Development* Executive Editor
Aaron Weule, IS Manager
Patrick Meader, *VBPJ* Associate Editor
Fred Perry, Conference Planner
Ferris Ferdon, DevX Advertising Director
Robert Scoble, Conference Planner
Michael Hollister, Art & Production Director

**TOP ROW (L to R):**
Tom Donoghue, Conference Sales Director
Andrew Nickels, Customer Service Manager
Ann ODonnell, Conference Manager
Andrew King, DevX Associate Producer
Kevin Strehlo, *Enterprise Development* Publisher & Editor in Chief
Monica McQueen, *Java Pro* Production Coordinator
Patrick Bonstead, Circulation Associate
Lynne Matthes, Group Publisher
Paul Spyksma, Publisher, FTP Advertising Network
Margaret Horoszko, Marketing Designer
Chris Kinsman, Vice President of Technology
Stephen Evanczuk, Vice President of Electronic Products
Lilly Gutierrez, Recruitment Advertising Manager

**NOT AVAILABLE FOR GROUP PHOTO:**
Brendy Bell, DevX Western Regional Manager
Brendan Coen, *VBPJ* Regional Account Manager
Ron Cordek, *Enterprise Development* Regional Sales Manager
Shana Deane, *Java Pro* Regional Sales Manager
Michael Long, *Enterprise Development* East Coast Advertising Manager
Christine Marquez, Production Manager
Cynthia Morgan, *Enterprise Development* Editor
Jane Myer, DevX Midwest Regional Manager
Ashley Stringfellow, Office Assistant
Mana Tominaga, DevX & *VBPJ* Editorial Assistant
Amanda Tyler, DevX Eastern Regional Manager
Lorenzo Valencia, Facilities Manager
Kent Wu, *Java Pro* Advertising Director

**Doing it right with The Cody Kid.™**




*"Your heterogenous environments are a lot like ours."*

# Develop, debug and deploy cross-platform network applications fast, with Distinct RPC Toolkits.

Introducing the Distinct ONC RPC/XDR Toolkits, the fastest way to bring your corporate applications to the Windows 95 and NT desktop. In addition to our toolkit for C and C++ developers, Distinct delivers the first Java-based RPC toolkit. Distinct ONC RPC/XDR Toolkit for Java puts RPC-based distributed programming right at Java programmers' fingertips. And because ours is the first toolkit of its kind to receive 100% pure Java certification, you can now write platform-independent distributed applications. So, whether you are porting the front end of your UNIX applications to the Windows platforms, or creating a completely new distributed application over a heterogeneous network, rely on Distinct RPC toolkits to help you do it faster and more reliably. It's also the easiest way ever devised to bring your UNIX-based applications to your desktop.

Our RPC Toolkits are backed by comprehensive technical support from Distinct, a pioneer in enabling technologies for software professionals since 1984. To get your free trial copy, visit the Cody Kid™ on our Web site today.

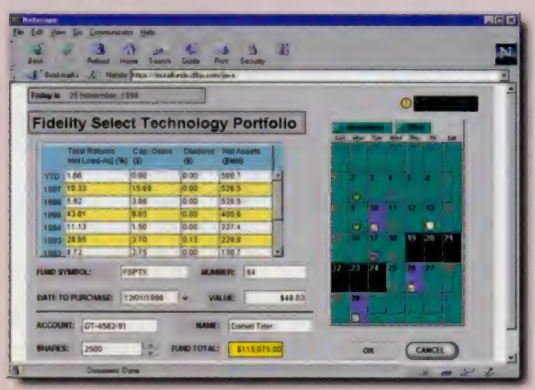
distinct
*Develop. Debug. Deploy.™*

**Download a trial version now at www.distinct.com or call 408-366-8933**

DISTINCT is a registered trademark and the DISTINCT logo, product names, the DEVELOP. DEBUG. DEPLOY. tagline, DOING IT RIGHT WITH THE CODY KID., and THE CODY KID are trademarks of Distinct Corporation. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc., in the United States and other countries. Windows and Windows NT are either trademarks or registered trademarks of Microsoft Corporation. All other trademarks are the property of their respective owners. ©1998 Distinct Corporation. 003

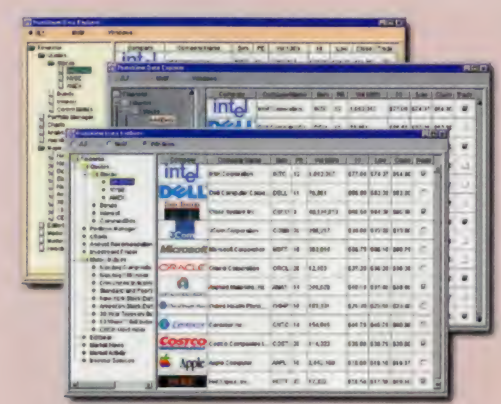
# Build It - Once

# Build It - Right




## ProtoView JFCSuite™

Powerful, extendable, lightweight components built on the foundation of JFC, the ProtoView JFCSuite contains JFCDataCalendar, JFCDataExplorer and JFCDataInput. JFCDataCalendar has drop-down date selection, multiple date & range selection and day-specific font/color/image properties. JFCDataExplorer provides an intuitive explorer UI by combining the JFC JTree and JTable into one integrated and enhanced package. Right-hand views can be customized to display JavaBeans components for each branch of the tree. JFCDataInput provides advanced validation and formatting for Currency, Date, Mask Edit, Numeric and Password input. Together, these components have everything you need to build solid, state-of-the-art, JFC based Java applications.



## ProtoView JSuite (AWT 1.1 JavaBeans)™

Winner of the JDJ World Class Award. The JSuite includes the DataTableJ grid component with JDBC and Visual Café database support. Also includes TreeViewJ, CalendarJ, TabJ and WinJ (MaskEdit, Currency, Numeric Edit, DateEdit Long, Time, Combo Box and many more).

**ProtoView Java Components** —— Download Free Trial Versions

Order Online Today!

**www.protoview.com**
**1-800-231-8588**

PROTOVIEW™
Software Tools for Component Based Development

Copyright © 1998 ProtoView Development Corporation. All rights reserved. All ProtoView product names are trademarks of ProtoView Development Corporation.

# JAVA Pro

**YES** ☑ Send me one year (12 issues) of *Java™ Pro* for only $29.95

Name _____

Address _____

City _____ State _____

Zip _____ Country _____

E-mail address _____

❑ Bill me ❑ Payment enclosed Charge my: ❑ Visa ❑ Master Card ❑ AMEX

Card # _____

Expiration date _____

Signature _____

Savings based on annual cover price of $71.40. Please allow 4-6 weeks for delivery of your first issue. International subscriptions must be payable in US dollars plus postage: Canada/Mexico add $18/year for surface delivery; all other countries add $44/year for airmail delivery.

4034

SAVE 58% SAVE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

**BUSINESS REPLY MAIL**

FIRST-CLASS MAIL PERMIT NO. 317 Palo Alto, CA

POSTAGE WILL BE PAID BY ADDRESSEE

**P.O. Box 54584**
**Boulder, CO 80323-4584**

# JAVA™ Pro Savings Voucher

Get *Java™ Pro* for less than $2.50 per issue—and save over 58% off the cover price. A single tip in one issue can more than pay for a year's subscription. Order Now and Save!

| Annual Cover Price |
| --- |
| $71.40 |

| Your Price |
| --- |
| $29.95 |

| Subscription Savings |
| --- |
| 58% |

❑ **1 Year** (12 issues) $29.95—SAVE 58%

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

❑ Payment enclosed ❑ Bill me later

One year is 12 monthly issues. Savings based on annual cover price of $71.40. Please allow 4-6 weeks for delivery of first issue. International subscriptions must be payable in U.S. dollars plus postage. Canada and Mexico add $18/year, all other countries add $44/year for airmail delivery. Canadian GST included.

**My Guarantee:**
If at any time I do not agree that *Java™ Pro* is the best, most useful source of information on Java Programming, I can cancel my subscription and recieve a full refund.

4032

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

**BUSINESS REPLY MAIL**

FIRST-CLASS MAIL PERMIT NO. 317 Palo Alto, CA

POSTAGE WILL BE PAID BY ADDRESSEE

**P.O. Box 54584**
**Boulder, CO 80323-4584**